Verwendung von Gamifizierung und Sensoren mobiler Endgeräte zur Absicherung von Webanwendungen gegen automatisierte Angriffe

Masterarbeit

zur Erlangung des akademischen Grades

Master of Science (M. Sc.)

vorgelegt an der Fachhochschule Köln

Campus Gummersbach

Fakultät für Informatik und Ingenieurwissenschaften

im Studiengang

Medieninformatik

Ausgearbeitet von:	Jens Kröner
Matrikelnummer:	11088185
Erster Prüfer:	Prof. Dr. Fischer
Zweiter Prüfer:	Prof. Dr. Karsch

Gummersbach, im Mai 2014

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Masterarbeit selbständig verfasst und gelieferte Datensätze, Zeichnungen, Skizzen und graphische Darstellungen selbständig erstellt habe. Ich habe keine anderen Quellen als die angegebenen benutzt und habe die Stellen der Arbeit, die anderen Werken entnommen sind - einschließlich verwendeter Tabellen und Abbildungen - in jedem einzelnen Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht.

Gummersbach, im Mai 2014

Jens Kröner

Kurzfassung

CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart), auch HIPs (Human Interaction Proof) genannt, sind weit verbreitete Sicherheitsmaßnahmen im World Wide Web, die Online-Dienste vor dem Missbrauch durch automatisierte Programme schützen sollen. Dazu werden die Benutzer aufgefordert, eine Aufgabe zu lösen, welche von Computern nicht gelöst werden kann, wie beispielsweise das Entziffern von verkrümmten Buchstaben. Durch die immer weiter verbesserten Fähigkeiten der Computerprogramme auf diesen Gebieten, wird es für die Entwickler der HIPs immer anspruchsvoller eine ausreichende Sicherheit zu garantieren. Daraus resultiert ein Zielkonflikt zwischen der Sicherheit und Gebrauchstauglichkeit der HIPs. Da die Sicherheit garantiert werden muss, leidet die Gebrauchstauglichkeit und die Benutzer sind oftmals nicht in der Lage das HIP zu lösen. Dies führte bereits in der Vergangenheit zu einer Abneigung gegenüber diesen Systemen.

In dieser Arbeit wurde überprüft ob diese Abneigung durch die Verwendung von Gamifizierung, also Spielelementen, in einem neu entwickelten HIP verringert werden kann. Zusätzlich sollte geprüft werden, ob anstelle der Tastatur die Sensoren mobiler Endgeräte als Eingabemedium dienen können. Dazu wurde durch eine Analyse von durchgeführten Angriffen auf bestehende HIPs sowie Studien zur Gebrauchstauglichkeit derselben ein Rahmenwerk erstellt. Dieses Rahmenwerk definiert Anforderungen, die ein sicheres und zugleich benutzerfreundliches HIP erfüllen muss. Anhand dieser Anforderungen wurde ein neues spielbasiertes HIP in Form eines Kugellabyrinths auf Basis von JavaScript und HTML5 implementiert. Der Benutzer muss durch die Neigung des Geräts eine Kugel steuern und diese innerhalb der vorgegebenen Zeit in das Loch lenken. Die Gebrauchstauglichkeit dieses HIPs konnte durch eine Onlineumfrage bestätigt werden. Dies gilt jedoch nicht für die Sicherheit, welche durch den lesbaren Quellcode stark beeinträchtigt wird. Für diese Schwachstelle konnte eine Lösung in Form einer Browsererweiterung durch eine Public-Key-Signatur konzipiert werden. Durch diese Erweiterung könnte auf den Einsatz von HIPs in vielen Bereichen verzichtet und andere Webanwendungen sicherer gestaltet werden.

Inhaltsverzeichnis

1 Einleitung

Im Jahr 2000 forschten an der Carnegie Mellon University in Pennsylvania Luis von Ahn, Manuel Blum, Nicholas Hopper und John Langford an einem System, mit dessen Hilfe bestimmte Bereiche des Internets gegen Angriffe von Maschinen geschützt werden sollten. Diese Untersuchung, die von Yahoo in Auftrag gegeben wurde, lieferte eine Technologie, welche die Erfinder *Completely Automated Public Turing Test to Tell Computers and Humans Apart* oder kurz CAPTCHA nannten [GWLY11]. Da es sich bei CAPTCHA um einen eingetragenen Markennamen der Carnegie Mellon University handelt [Pon06], wird in dieser Ausarbeitung der Begriff *Human Interaction Proof* (HIP) verwendet.

Anlass dieser Forschung war die Tatsache, dass Yahoo's Live-Chats von automatisierten Angriffen überrannt wurden. Dabei versuchten Bots[1] die Benutzer der Chats durch Links auf Webseiten mit Werbung zu locken [HCR09]. Da Webanwendungen, wie in diesem Fall die Chats, nur von Menschen genutzt werden sollen, wird durch den Einsatz von Bots das IT-Schutzziel der Authentizität verletzt, es kann hierbei also von einem Angriff gesprochen werden.

Im Laufe der Zeit wurden diese HIPs in immer mehr Bereichen eingesetzt, die vor einem automatisierten Zugriff geschützt werden sollen. Dieser Zugriff kann beispielsweise durch Bots erfolgen, die pro Minute tausende neue E-Mail Konten erstellen und anschließend die vielfache Menge von Spam E-Mails versenden können [Yan06, GWLY11, OM12]. Darüber hinaus gibt es Bots, die Foren mit Beiträgen oder Artikel mit Kommentaren überschwemmen. Diese enthalten Links, die arglose Benutzer meist auf spezielle Internetseiten leiten, um dort deren Geräte mit Viren oder Trojanern zu infizieren. Zudem existieren Suchmaschinen-Bots, die automatisch Webseiten registrieren um das Ranking in anderen Suchmaschinen zu erhöhen [Yan06]. Ebenso können die Crawler von Suchmaschinen auf diese Weise von privaten Seiten im Internet abgehalten werden [GWLY11]. Filehoster wie Mega (früher Megaupload) und Hotfile schützen sich mit HIPs gegen das automatisierte Hoch- bzw. Herunterladen von Dateien [OM12]. Und Onlineshops können damit verhindern, dass massenhaft Artikel gekauft werden um diese anschließend wieder teurer zu veräußern. Diese Praxis ist vor allem bei Veranstaltungstickets gebräuchlich [OM12]. Ein anderer Gesichtspunkt ist, dass die Finanzierung vieler Internetauftritte durch geschaltete Werbung erfolgt. Werden die Inhalte dieser Seiten jedoch von Maschinen abgegriffen und dem Benutzer anschließend werbefrei präsentiert, führt dies zu geringeren Einnahmen und deshalb zum Schutz vor solchen Vorgängen durch HIPs [Sut12].

Die Technik der HIPs ist an den Turing Test angelehnt. Dieser von Alan Turing 1950 vorgeschlagene Test soll es einem Menschen ermöglichen, durch ein Gespräch mit zwei Kommunikationspartnern, einer Maschine und einem Menschen, die Maschine zu enttarnen [Tur50]. Da es sich bei dem aktuellen Problem um einen Computer handelt, der versucht herauszufinden ob sein Gegenüber ein Mensch oder ein Computer ist, wird dieses Vorhaben in der Literatur auch oftmals als *reverse turing test*, also umgekehrter Turing Test, bezeichnet. Eine weitere Abweichung findet sich in der Dauer des Gespräches. Während diese von Turing mit fünf Minuten angesetzt war, muss heute eine einzige Aufgabe diesen Zweck erfüllen [Tur50]. Zudem sollte diese Aufgabe für den Benutzer innerhalb kürzester Zeit lösbar sein, da dieser sonst frustriert aufgeben könnte und den Dienst möglicherweise nicht mehr nutzt. Die

[1] Ein Bot ist ein Computerprogramm, das wiederkehrende Aufgaben automatisiert erledigen kann (vgl. http://de.wikipedia.org/wiki/Bot).

Grundidee hinter HIPs ist deshalb, eine Aufgabe zu stellen, die von Menschen leicht, von Computern jedoch nur sehr schwer bzw. nach langer Rechenzeit oder mit hohem Aufwand korrekt gelöst werden kann [YSEA08b].

Leider gestaltet sich die Umsetzung dieses Verfahrens alles andere als einfach. So können viele dieser HIPs von Menschen mit Behinderung nicht immer korrekt gelöst werden [GWLY11]. Dazu zählen zum einen Blinde, welche die visuell gestellte Aufgabe auf dem Bildschirm überhaupt nicht sehen können [YSEA08b]. Da es sich um eine Aufgabe handelt, die von Computern schwer zu lösen ist, können auch die Screen Reader, welche häufig von Blinden eingesetzt werden um den Bildschirminhalt erfassen zu können, diese nicht entziffern. Zum anderen sind davon auch Menschen mit anderen Sehbehinderungen und Analphabeten betroffen [YSEA08b]. Um diese Menschen nicht auszuschließen wurden audiobasierte HIPs entwickelt. Leider ist die große Mehrheit dieser HIPs sowohl für Blinde als auch für Sehende nur mit sehr großen Anstrengungen lösbar, was in Kapitel 4 näher beleuchtet wird [BC09, RC13].

Bis heute wurde noch keine Untersuchung durchgeführt, die besagt, dass Menschen überhaupt in der Lage sind, HIPs zu lösen [NB12]. Die Entwicklung immer neuerer Verfahren beruht auf Annahmen und Versuchen, die im *trial and error*[2] Vorgehen praktiziert wurden [YSEA07]. So ist nicht klar, welche Eigenschaften der Sicherheit oder der Gebrauchstauglichkeit dienlich sind. Das bedeutet auch, dass keine fundierte Aussage über die Gebrauchstauglichkeit und die Sicherheit der HIPs gegeben werden kann. Es fehlt demnach eine Grundlage, die Auskunft darüber geben kann, welche Art von visuellen Rätseln von Menschen in angemessenem Umfang rezipiert werden können und die gleichzeitig eine angemessene Hürde für die künstliche Intelligenz darstellen. Da die Fähigkeiten der Computerprogramme auf diesen Gebieten ständig verbessert wurden, entstand ein Wettlauf zwischen den Angreifern und den Entwicklern dieser HIPs.

Zudem bringt die Verwendung von HIPs weitere Nachteile mit sich [GWLY11]: Um eine ausreichende Sicherheit gegenüber dem schlichten Raten eines Computers gewährleisten zu können, bedarf es einer großen Rätselbibliothek. Aus Sicherheitsgründen empfiehlt sich ein separater Server, der die HIPs bereitstellt und die Antworten der Benutzer überprüft. Dieses Vorgehen ist allerdings mit Mehrkosten für die benötigte Hard- und Software verbunden. Des Weiteren steigt auch das Datenvolumen, das zwischen dem Benutzer und dem Anbieter ausgetauscht wird. Die wohl bedeutsamsten Faktoren sind die Zeit und Arbeit, die der Benutzer aufbringen muss um das HIP zu lösen. Übersteigen diese den Nutzen des Internetangebots, hat der Anbieter mit sinkenden Benutzerzahlen zu kämpfen [GWLY11]. Es muss demnach ein Gleichgewicht zwischen Aufwand, Gebrauchstauglichkeit und zugleich Sicherheit gefunden werden.

Viele der eingesetzten HIPs sollen den Seitenbetreiber vor Spam oder automatisierten Suchanfragen schützen. Dies ist jedoch nicht das Problem des Benutzers, welcher aus diesem Grund nicht mit seiner Zeit dafür bezahlen sollte [Bri11]. Tatsächlich kann der Einsatz eines HIPs in manchen Fällen überflüssig sein. So können beispielsweise Onlineformulare durch die Kombination von Honeypot Feldern, Zeitstempel, dynamischer Feldnamen und Eingabefilter geschützt werden [Mot13, Tul], was in Abschnitt 2.6 genauer erläutert wird. Von diesen Maßnahmen merkt der normale Benutzer nichts und wird nicht davon belästigt. Für den Seitenbetreiber kann dies wiederum positiv sein: Animoto, ein Onlinedienst, bei dem Benutzer Bilder hochladen können um daraus ein Video zu generieren, gewann im Jahr 2009 durchschnittlich 2400 neue Benutzer pro Tag. Das Entfernen des reCAPTCHA HIPs aus dem Registrationsformular steigerte die täglichen Neuanmeldungen um 33% [Bri11].

[2] Bei *trial and error* oder Versuch und Irrtum wird versucht ein Problem durch Ausprobieren zu lösen (vgl. http://de.wikipedia.org/wiki/Versuch_und_Irrtum).

Ein weiterer Aspekt in Sachen Kundenzufriedenheit ist die Tatsache, dass die meisten der heute eingesetzten HIPs nicht für den Einsatz auf mobilen Endgeräten entwickelt wurden. So kann es bei textbasierten HIPs passieren, dass das HIP beim fokussieren der Antwortbox nicht mehr sichtbar ist. Da die mobile Nutzung des Internets auch weiterhin zunehmen wird, ist es wichtig für diesen Fall eine angemessene Lösung vorweisen zu können [D2113, LHBL11].

Durch die Anforderungen an die Sicherheit und zugleich Gebrauchstauglichkeit der HIPs entsteht ein Zielkonflikt, den es zu lösen gilt. Deshalb wurden in dieser Arbeit durch die Analyse der veröffentlichten Angriffe auf verschiedenste HIPs, und Studien über die Gebrauchstauglichkeit Sicherheitslücken und Schwachpunkte aufgedeckt und zu Richtlinien für die Entwicklung eines sicheren und benutzerfreundlichen HIPs zusammengefasst. Zudem wurde der Versuch unternommen, Teile hiervon in einer Studie umzusetzen und ein HIP für den Einsatz auf mobilen Endgeräten zu entwickeln.

Zahlreiche Untersuchungen konnten zeigen, dass mittels Gamifizierung eine erhöhte Produktivität durch intrinsische Motivation erreicht werden kann [DDKN11, Ham13, HKS14]. Gamifizierung, selten auch Spielifizierung genannt, bezeichnet den Einsatz von Spielelementen in Nicht-Spiel-Kontexten [DDKN11]. Dadurch kann beispielsweise eine Aufgabe des Arbeitsalltags durch die Anzeige von gesammelten Punkten und einer Rangliste motivierender gestaltet werden [DDKN11]. Für die Neuentwicklung im Rahmen dieser Arbeit wird Gamifizierung eingesetzt, um das ansonsten lästige Lösen von HIPs interessanter zu gestalten, und dem Benutzer möglicherweise sogar Freude zu bereiten. Um die Gebrauchstauglichkeit zusätzlich zu steigern, verwendet dieser Ansatz als Eingabemedium anstelle der Bildschirmtastatur die verbauten Lagesensoren der mobilen Endgeräte.

Im folgenden Kapitel wird ein chronologischer Überblick über die Entwicklung der HIPs gegeben und die verschiedenen HIP Techniken, die im Laufe der Zeit entwickelt wurden, näher beschrieben. In Kapitel 3 wird anhand veröffentlichter Angriffe gezeigt, wie diffizil die Entwicklung eines sicheren HIPs ist und welche Aspekte dabei betrachtet werden sollten. Aus den dort ausgenutzten Sicherheitslücken wurden Eigenschaften abgeleitet, die von einem HIP erfüllt werden müssen, soll es ein ausreichendes Maß an Sicherheit bieten. Kapitel 4 behandelt die Schwachstellen verschiedener HIPs im Hinblick auf die Gebrauchstauglichkeit. Daraus wurden ebenfalls Verbesserungsmöglichkeiten abgeleitet und zusammengefasst. Die Studie, die im Rahmen dieser Arbeit durchgeführt wurde und versucht durch alternative Eingabemöglichkeiten und Gamifizierung die Gebrauchstauglichkeit von HIPs auf mobilen Endgeräten wesentlich zu verbessern, wird in Kapitel 5 behandelt. Das Fazit wird in Kapitel 6 gezogen, darin erfolgt zudem ein Ausblick über die mögliche Weiterentwicklung von HIPs und des im Rahmen dieser Arbeit entwickelten Prototypen.

2 HIP Techniken

In diesem Kapitel werden verschiedene Formen von HIPs dargestellt. Dabei kann zwischen text-, bild-, audio-, video- und spielbasierten HIPs unterschieden werden. Mit diesem Kapitel soll ein Überblick über die Methodiken der HIPs und erste tiefere Einblicke in die Funktionsweise und Technik hinter den Kulissen gegeben werden. Dazu folgt in diesem Abschnitt die chronologische Entwicklung der verschiedenen Techniken. Diese werden in den darauf folgenden Abschnitten anhand von Beispielen genauer erläutert und beschrieben. Daran schließen sich je ein Abschnitt über die grundlegende Methodik der HIPs und eine Zusammenfassung an.

Egal welche Technik für ein HIP verwendet wird, die Erzeugung des HIPs muss so einfach sein, dass sie von einem Computer allein vorgenommen werden kann [MMH09]. Ein HIP kann deshalb auch mit einer Einwegfunktion eines Verschlüsselungsalgorithmus verglichen werden: Ein Computer generiert Daten nach einem vorgegebenen Ablauf, während ein anderer Computer nicht in der Lage sein soll diese Daten zu interpretieren. Nur ein Mensch sollte diese Aufgabe aufgrund seiner kognitiven Fähigkeiten lösen können. Um diese Anforderungen zu erfüllen ist es notwendig, dass HIPs für Menschen leicht les- oder erkennbar sind, jedoch die Lösung nicht erraten werden kann [HGH08].

Auch wenn der Begriff des CAPTCHA von der Entwicklung an der Carnegie Mellon University im Jahr 2000 geprägt wurde, kam diese Technologie bereits vorher zum Einsatz. So arbeitete man bei AltaVista bereits im Jahr 1997 an einer Absicherung der Suchmaschine gegen die Beeinflussung durch Bots [Klu08c]. Dort entwickelten Lillibridge et al. ein Verfahren, um Bots durch verzerrten Text abwehren zu können und meldeten dies als Patent an [LABB98]. Da das Patent jedoch erst im Jahr 2001 veröffentlicht wurde, herrsche Unklarheit darüber wer CAPTCHAs wirklich erfunden hatte. Die Erfinder orientierten sich während der Entwicklung an einem Handbuch eines Scanners [LABB98]. Darin waren Empfehlungen aufgeführt, wie Texte für *Optical Character Recognition* (OCR) optimiert werden können. Um das HIP zu entwickeln, kehrten sie diese in das genaue Gegenteil um [Klu08c].

In den folgenden Jahren stieg die Popularität der textbasierten HIPs immer weiter. Dies hatte zur Folge, dass Angreifer, die weiterhin Dienste für die Verbreitung von Spam, Werbung oder Phishing Attacken benötigten, auf Seiten auswichen, die keine HIPs verwendeten. Mit der Zeit wurden immer mehr Angriffsmöglichkeiten auf die HIPs selbst bekannt. Im Jahr 2003 konnten manche HIPs zu 92% geknackt werden [MM03]. Schon damals wurde klar, dass die eigentliche Schwierigkeit darin liegt, die einzelnen Buchstaben aus dem Bild zu extrahieren, das Bild also zu segmentieren. Aus diesem Grund gilt ein textbasiertes HIP als gebrochen, sobald es in die einzelnen Zeichen zerlegt werden kann [YSEA08a]. Daraus resultierten nicht etwa neue Modelle solcher HIPs, sondern es wurde versucht diese Angriffe durch überarbeiten der Implementierung zu verbessern [HCR09].

Im Jahr 2009 wurde von Megaupload ein HIP verwendet, das durch eine sehr starke Überlappung der einzelnen Zeichen vor der Segmentierung sicher sein sollte (siehe Abbildung 2.1(a)) [HCR09]. reCAPTCHA verwendete 2010 ein ähnliches Verfahren um die Segmentierung zu verhindern. Dabei wurden die Zeichen nicht ineinander geschrieben, sondern einige der Zeichen mit einem schwarzen Kreis hinterlegt und die Farbe der Buchstaben in diesem Bereich invertiert, wie es in Abbildung 2.1(b) zu sehen ist [SEAYM10].

(a) Beispiel eines Megaupload HIPs (2009).

(b) Beispiel eines reCAPTCHA HIPs (2010).

Abbildung 2.1: Diese HIPs setzten auf eine besondere Technik, welche die Segmentierung verhindern sollte (aus [SEAYM10]).

Bereits wenige Jahre nach den ersten textbasierten wurden bildbasierte HIPs entwickelt. Abbildung 2.2 zeigt ein Beispiel des ersten, von Chew und Tygar [CT04] entwickelten, bildbasierten HIPs [HCR09]. Bis heute wurden viele weitere verschiedene Ansätze dieser HIPs veröffentlicht, von denen einige in Abschnitt 2.3 näher beschrieben werden.

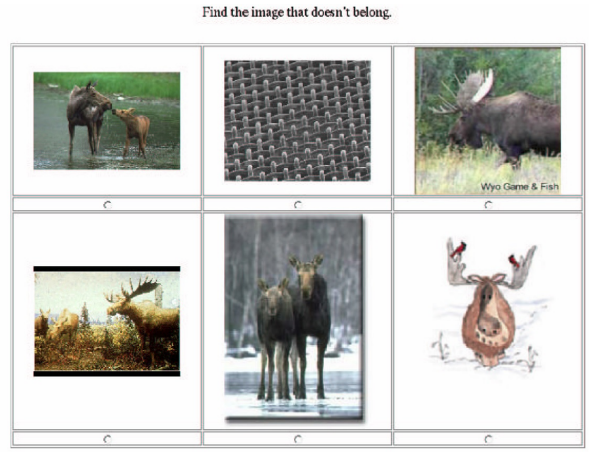


Abbildung 2.2: Ein Beispiel des von Chew und Tygar [CT04] veröffentlichten HIPs. In diesem Fall muss jenes Bild ausgewählt werden, das nicht zu den anderen passt.

Ein Problem der meisten bildbasierten Ansätze ist jedoch eine zu geringe Anzahl von verwendeten Bildern. KittenAuth verwendete Bilder von Tierbabys um Computer von Menschen zu unterscheiden. Aus 9 Bildern mussten die 3 Bilder herausgesucht werden, die kleine Katzen zeigten. Jedoch wies die gesamte Bilddatenbank weniger als 100 Bilder auf [HCR09]. Eine so geringe Anzahl kann binnen weniger Minuten klassifiziert werden. Danach kann jedes Exemplar dieses HIPs vom Computer gelöst werden. Dieses Problem wurde bei HumanAuth versucht zu lösen: Durch Wasserzeichen, die zufällig im Bild platziert wurden, sollten die Bilder sich ausreichend unterscheiden um nicht erkannt werden zu können, was jedoch nicht gelang [HCR09]. Besser scheint hierbei Asirra zu sein, dessen Bilddatenbank 3 Millionen Bilder umfasst [EDHS07]. Dieses HIP wird in Abschnitt 2.3 näher beschrieben.

Bereits 2002 wurden audiobasierte HIPs entwickelt, die von sehbehinderten Benutzern leichter zu lösen sein sollten, was jedoch nicht in allen Fällen gegeben ist [KLS02, BC09, RC13]. Während es sich dabei um eigenständige HIPs handelte, werden diese heute meist als Alternative zu den textbasierten HIPs angeboten [BC09]. Die Wiedergabe des Audiosamples kann in einem solchen Fall durch eine Schaltfläche gestartet werden. Jedoch wurden die ersten Versionen der Audio-HIPs von Google, Microsoft und Yahoo schon nach kurzer Zeit geknackt [BBP+11]. Diese basierten darauf, dass eine Reihe von Ziffern gesprochen wurde. Um zu verhindern, dass Computerprogramme die Ziffern identifizieren können, wurde das Audiosignal verzerrt und durch Rauschen verfälscht. Der neueste Ansatz ist die Verwendung von *semantic noise*, was in Abschnitt 2.2 genauer beschrieben wird.

Aktuell werden immer mehr spielbasierte HIPs entwickelt. Diese haben den Vorteil, dass die Aufgabe komplexer sein kann als dies beispielsweise bei textbasierten HIPs der Fall ist. Während dort lediglich das Erkennen und Wiedergeben von Zeichen verlangt ist, so müssen hier unter anderem komplexe Formen erkannt werden und Gegenstände an vordefinierten Positionen abgelegt werden. Dies erschwert einerseits dem Angreifer das Brechen des HIPs und andererseits bietet es dem Benutzer eine angenehmere Art der Unterhaltung.

In den folgenden Abschnitten werden für die hier genannten Techniken einige Beispiele aufgeführt. Damit soll ein erster Eindruck der Vielfalt gegeben werden, die mittlerweile in diesem Umfeld existiert. Falls die gezeigten Beispiele heute nicht mehr verwendet werden, ist dies an entsprechender Stelle angegeben. Im Anschluss daran wird die grundlegende Methodik beschrieben, die den Ablauf der Authentifizierung ermöglicht. Dabei geht es vor allem um die Kommunikation zwischen dem Server, welcher das HIP bereitstellt, und dem Gerät des Benutzers. Am Ende erfolgt eine Zusammenfassung dieses Kapitels.

2.1 Textbasiert

Diese Technik des HIPs wird bis heute am häufigsten genutzt [YSEA08b, Sut12]. Hierbei wird die Schwäche der Computer ausgenutzt, deformierte und verschobene Zeichen in Bildern zu erkennen. Auch wenn Text in Form von Bildern verwendet wird, zählen diese HIPs zu den Textbasierten. Wird ein solches HIP korrekt entworfen, so ist es sicher gegen *brute force* Angriffe [YSEA08a]. Werden jedoch nur wenige verschiedene oder eine feste Anzahl von Zeichen verwendet, sinkt die Zahl der Kombinationsmöglichkeiten stark ab. In Extremfällen kann dann durch Raten ein HIP korrekt gelöst werden. Abbildung 2.3 zeigt eine kleine Auswahl textbasierter HIPs.

In den letzten Jahren wurde vermehrt auf Farben gesetzt und sowohl die Schrift als auch der Hintergrund unterschiedlich eingefärbt [JS12]. Oftmals wurde für den Hintergrund ein chaotisches Muster von vielen verschiedenen Farben verwendet, wie in Abbildung 2.3(d) zu sehen. Leider litt hierunter nur die Lesbarkeit für den Menschen, während Computer diese Farben sehr leicht ausblenden konnten. In wenigen Fällen verbesserte dies sogar die Lesbarkeit für den Computer [YSEA08b]. Außerdem ist es problematisch wenn immer derselbe oder nur wenige unterschiedliche Hintergründe verwendet werden. Denn diese Hintergründe sind nach wenigen Durchläufen bekannt und können einfach entfernt werden [HGH08].

In den meisten textbasierten HIPs werden Bitmap-Schriften[1] eingesetzt. Die Platzierung der Zeichen an festen Positionen stellt nur eine kleine Hürde für Bots dar, während zufällige Platzierungen sowohl die Segmentierung als auch das Identifizieren der Zeichen erschweren kann. Bei einer zufälligen Platzierung muss jedoch darauf geachtet werden, dass die Reihenfolge der Zeichen stets klar erkennbar ist. Dabei darf kein Zeichen aus der Reihe fallen [HGH08]. Auch wenn häufig Buchstaben in Groß- und Kleinschreibung verwendet werden, so wird die Eingabe des Benutzers nur in den seltensten Fällen dahingehend geprüft.

Oft wird Rauschen in Form von Pixeln, Linien oder anderen Objekten verwendet, wie es das PayPal HIP in Abbildung 2.3(a) zeigt. Dabei kann zwischen Vorder- und Hintergrundrauschen unterschieden werden, denn Vordergrundrauschen berührt die Zeichen, im Gegensatz zum Hintergrundrauschen. Um die Identifikation der Zeichen zu erschweren kann eine lineare Transformation durchgeführt

[1] Bitmap-Schriften liegen als Rastergrafiken vor und können deshalb nur bedingt vergrößert werden. Dies erleichtert jedoch die Bearbeitung, die bei diesen HIPs häufig angewendet wird.

werden. Dabei werden Zeichen rotiert oder verdreht. Fortschrittlichere HIPs verwenden dagegen die nicht-lineare Transformation sowie gekrümmte Zeichen, da diese noch schwerer zu identifizieren sind. Bei einigen HIPs kommen ausgefallene Methoden für die Verzerrung der Zeichen zum Einsatz, dies erschwert zwar einerseits die Identifikation, andererseits jedoch auch die Lesbarkeit [HGH08].

Bei dem in Abbildung 2.3(d) gezeigten Beispiel eines Gimpy HIPs mussten 3 der 7 gezeigten Wörter entziffert werden. Dies ermöglichte dem Computer eine viel größere Chance Worte zu finden, die er entziffern konnte. Aus diesem Grund wird dieses HIP heute nicht mehr verwendet [JS12].

Ein anderer Ansatz wird in Abbildung 2.3(e) gezeigt. Hier wurde die Schwäche der Texterkennung in Bezug auf Handschriften ausgenutzt. Da diese Handschriften jedoch synthetisch generiert sind, ist auch die Lesbarkeit durch den Menschen nicht besonders hoch und dieses HIP findet deshalb keinen Einsatz mehr [JS12].

Die Abbildungen 2.3(b) und 2.3(c) zeigen HIPs, die in dieser oder in leicht abgewandelter Form heute noch verwendet werden. reCAPTCHA stellt hierbei einen interessanten Mix aus einer normalen text- und einer datenbankbasierten Aufgabe dar. So werden hierbei zwei Wörter abgefragt. Das linke Wort in Abbildung 2.3(c) stellt die eigentliche Aufgabe dar, und ist deshalb in den meisten Fällen auch schwerer zu entziffern [HCR09]. Das rechte Wort stammt dagegen aus dem Texterkennungsprozess von Google Books, und konnte dort nicht korrekt erkannt werden. Selten verwendet dieses HIP auch Hausnummern, die im Google Street View Projekt gesammelt und nicht erkannt werden konnten. Damit machen sich die Entwickler die Tatsache zunutze, dass das Wort bereits von einem OCR Prozess nicht korrekt erkannt werden konnte. Stimmt die Antwort mehrerer Benutzer für dieses Wort überein, so wird dies an den Texterkennungsprozess zurückgegeben [reC04]. Da dieses Wort auch dem HIP unbekannt ist, muss hierfür nicht zwingend etwas eingegeben werden. Die korrekte Antwort auf das gezeigte HIP kann also schlicht „ctteeze“ lauten.

(a) Ein solches HIP wurde 2008 von PayPal verwendet (aus [HHY12]).

(b) Gmail verwendet bis heute derartige HIPs (aus [HHY12]).

(c) Ein solches HIP wird bis heute von reCAPTCHA verwendet (aus [reC04]).

(d) Das Gimpy HIP wurde 2003 von Yahoo eingesetzt (aus [JS12]).

(e) Synthetische Handschrift, die als HIP verwendet wurde (aus [HHY12]).

Abbildung 2.3: Verschiedene textbasierte HIPs.

2.2 Audiobasiert

Audio-HIPs werden oft nur als eine Alternative zu den Textbasierten angeboten. Auch hier gibt es jedoch Schwierigkeiten: Wenn der Benutzer sich in einer lauten Umgebung befindet, kann er das Audiosample nicht verstehen. Zudem sind diese HIPs von blinden Benutzern, für die sie eigentlich entwickelt worden waren, nur schwer zu benutzen [BAB+08]. Dies liegt zum einen am Design des Interfaces, zum anderen aber auch an den sehr schwer zu verstehenden Audiosamples [BC09, RC13]. Diese Aspekte werden in Kapitel 4 genauer behandelt.

In anderen Fällen wird eine konkrete Frage als Audiosample bereitgestellt, die der Benutzer dann durch eine Tastatureingabe beantworten muss [JS12]. Dieses Vorgehen ähnelt stark dem von Video HIPs, die im nächsten Abschnitt beschrieben werden.

Wie auch die textbasierten HIPs beruht dieser Ansatz darauf, dass die menschliche Audiorezeption dem Computer voraus ist. Da hier jedoch in letzter Zeit enorme Fortschritte gemacht wurden, muss das Audiosample derart bearbeitet werden, dass die Erfolgsquote auch beim Menschen extrem gesunken ist [BBP+11]. Um Maschinen das identifizieren von Störgeräuschen zu erschweren, wird aktuell immer mehr auf *semantic noise* gesetzt. Dabei handelt es sich um Geräusche, die zum einen die gleiche Lautstärke, die gleiche Länge und ähnliche phonetische Eigenschaften wie die gesprochen Zeichen aufweisen. Dadurch können diese Geräusche nur auf einer semantischen Ebene korrekt als Störgeräusche erkannt werden [TSHA08]. Abbildung 2.4 zeigt einen drei Sekunden langen Ausschnitt des reCAPTCHA Audio-HIPs, bei dem nur Ziffern gesucht waren. Der zweite Ausschlag der Amplitude gegen Ende des Ausschnittes, der keine Beschriftung erhielt, ist ein Beispiel für *semantic noise* und enthielt ein „eee“. Es muss also zuerst verstanden werden was gesprochen wurde, um zu wissen, dass es sich dabei um keine Ziffer sondern um ein Störgeräusch handelt.

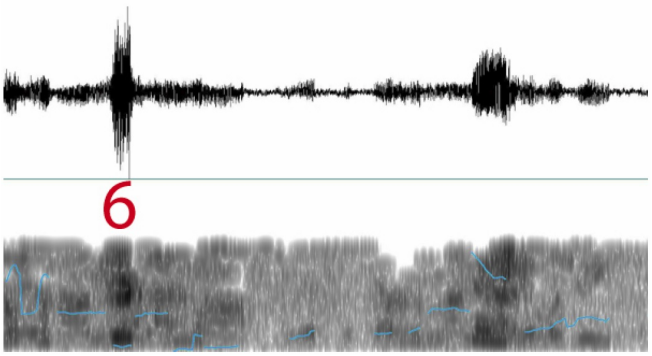


Abbildung 2.4: Ausschnitt eines reCAPTCHA Audio-HIPs, in dem die Amplitude einer gültigen Ziffer, sowie von *semantic noise* zu sehen ist (aus [BBF+10]).

2.3 Bildbasiert

Diese Ansätze machen sich die Schwächen der Bilderkennung zunutze, die Computer auch heute noch besitzen. So wird hier nicht versucht einen Text bis zur Unkenntlichkeit zu bearbeiten, sondern es werden meist einfache Fotografien verwendet. Dies macht diese Technik für die Menschen komfortabler, da die Bilder in den meisten Fällen nicht besonders bearbeitet wurden und deren Inhalt einfacher zu erkennen ist. Wie bereits zu Beginn dieses Kapitels beschrieben, ist die große Schwierigkeit hierbei, eine angemessen große Bilddatenbank zu verwenden. Deshalb kommen bildbasierte HIPs im Allgemeinen

sehr selten zum Einsatz. Ist dies der Fall, so handelt es sich hierbei um generierte Bilder, wie in Abbildung 2.5 zu sehen. Bei diesem HIP muss der nicht geschlossene Kreis gefunden und angeklickt werden. Diese Aufgabe wird durch die beiden schwarzen Balken erschwert, die sich abwechselnd über die gesamte Höhe des Bildes erstrecken.

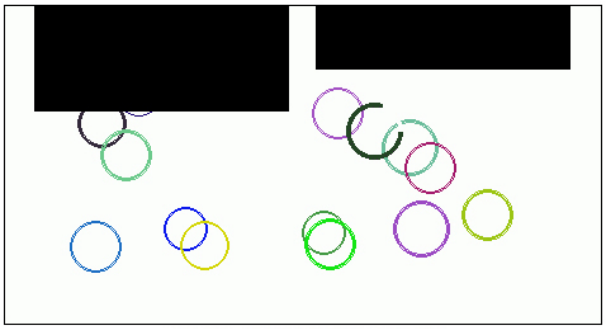


Abbildung 2.5: Ein generiertes bildbasiertes HIP, bei dem der nicht geschlossene Kreis gefunden werden muss (aus [Lin]).

Ein Beispiel für ein HIP, bei dem Fotografien verwendet wurden, ist Asirra (Animal Species Image Recognition for Restricting Access), das von Elson et al. [EDHS07] entwickelt wurde und Bilder von 12 Katzen und Hunden zeigt. Um das HIP erfolgreich zu lösen, muss der Benutzer alle Bilder auswählen auf denen Katzen zu sehen sind. Die Bilder stammen aus der Datenbank von Petfinder.com, die mehr als 3 Millionen vorsortierter Bilder umfasst. Bei Petfinder.com können zur Adoption freigegebene Haustiere adoptiert werden. Wird mit dem Mauszeiger über ein Bild gefahren, so wird dieses in vergrößerter Darstellung angezeigt, wie in Abbildung 2.6 zu sehen ist. Als Gegenleistung für die Nutzung der Bilddatenbank wird unter jedem Bild ein Link angezeigt, unter dem weitere Informationen über das Tier sowie die Möglichkeit gegeben ist dieses zu adoptieren [EDHS07, Gol08].

Abbildung 2.6: Bei Asirra müssen aus 12 Bildern alle diejenigen mit Katzen gefunden werden (aus [EDHS07]).

In Abbildung 2.7 ist ein Beispiel des HIPs KittenAuth gezeigt. Hierbei muss der Benutzer für ein erfolgreiches Lösen dieses HIPs drei Bilder von Katzen auswählen. Bei den anderen sechs Bildern handelt es sich um ähnlich aussehende Tiere wie Hasen, Eichhörnchen oder kleine Hunde [GWLY11]. Leider wurde keine Aussage darüber gemacht wie groß der Pool der Bilddaten ist.

Abbildung 2.7: Bei KittenAuth müssen drei Bilder von Katzen gefunden werden (aus [GWLY11]).

Bei Collage werden zufällig ausgewählte Cliparts in zufälligen Skalierungen, Drehungen und Positionen angeordnet. Ein Beispiel hierfür ist in Abbildung 2.8(a) zu sehen. Ein zusätzlicher Text gibt an, welches Bild vom Benutzer gefunden und angeklickt werden muss [JS12].

Ähnlich ist das Vorgehen von Implicit. Hier gibt der Text ein Wort vor, das gefunden und angeklickt werden muss, wie in Abbildung 2.8(b) gezeigt. Dieses Verfahren ist dadurch sicherer, da das Bild für eine Texterkennung erst aufgeteilt werden müsste [JS12].

(a) Collage (aus [JS12]). (b) Implicit (aus [JS12]).

Abbildung 2.8: Bildbasierte HIPs, die generierte Bilder verwenden.

Genau umgekehrt ist die Vorgehensweise von Multi Model, wie in Abbildung 2.9(a) zu sehen ist. Hier zeigt ein großes Bild den gesuchten Begriff, der nun, ebenfalls in das Bild integriert, durch einen Mausklick ausgewählt werden muss. Dieses HIP konnte bereits gebrochen werden und wird nicht mehr verwendet [RTD12].

ESPPix zeigt dem Benutzer eine bestimmte Anzahl an Bildern. Die Gemeinsamkeit der Bilder muss anschließend über eine Combo-Box ausgewählt werden. Trifft der Benutzer die richtige Auswahl, erhält er Zugang zum geschützten Bereich des Internetauftritts [RTD12]. Ein Beispiel hierfür ist in Abbildung 2.9(b) zu sehen, in welcher die Lösung *Ant* für Ameise bereits ausgewählt wurde. Ein Problem dieses HIPs ist sicherlich das Bereitstellen von ausreichend vielen Bildern.

Wie bereits im vorangegangen Abschnitt beschrieben, basieren wenige HIPs darauf, dem Benutzer ein kurzes Video anzuzeigen. Nachdem das Video abgespielt wurde, muss der Benutzer in wenigen Worten eingeben, was in dem Video gezeigt wurde. Da es hier aber sehr viel Raum für Interpretationen gibt, ist die Gebrauchstauglichkeit solcher HIPs eher gering [JS12].

(a) Multimodel (aus [RTD12]). (b) ESPPix (aus [RTD12]).

Abbildung 2.9: Bildbasierte HIPs, die Fotografien als Bilder verwenden.

2.4 Spielbasiert

Spielbasierte HIPs werden erst seit kurzer Zeit entwickelt, scheinen jedoch eine gelungene Alternative zu den sonst lästigen Aufgaben zu sein. Da keine wissenschaftlichen Arbeiten zu derzeit verwendeten spielbasierten HIPs gefunden werden konnten, wurden für die Erläuterungen in diesem Abschnitt ausschließlich kommerzielle Ansätze ausgewählt.

Das Detroiter Startup Unternehmen Are You a Human wurde im Jahr 2007 von Tyler Paxton gegründet. Motiviert wurde Paxton durch einen Vorfall, bei dem Hacker mit Bots alle Eintrittskarten eines Hannah Montana Konzertes aufkauften, um diese später zu weit höheren Preisen wieder zu veräußern. Zwar hatte die Ticketplattform versucht sich gegen derartige Angriffe mit textbasierten HIPs zu schützen, doch boten diese keinen ausreichenden Schutz [Are14c].

Paxton erkannte, dass die Erhöhung der Sicherheit von textbasierten HIPs diese gleichzeitig für Menschen unlösbar machen würde. Aus diesem Grund bietet Are You a Human HIPs in Form von Spielen an, bei denen der Benutzer verschiedene, sich zufällig bewegende Gegenstände in einen

vorgegeben Bereich verschieben muss. Abbildung 2.10 zeigt ein sogenanntes Are You a Human PlayThru, in welchem der Benutzer die korrekten Samen in das dafür vorgesehene Beet streuen muss. Während der Benutzer dies versucht, sammelt das Spiel alle Bewegungsdaten der Computermaus und speichert diese. Sind alle Gegenstände korrekt platziert, beendet sich das Spiel automatisch. In diesem Moment werden die gesammelten Daten an den Server von Are You a Human übertragen. Zu diesem Zeitpunkt ist bereits klar, dass das Spiel erfolgreich gelöst wurde, der Benutzer also alle erforderlichen Gegenstände korrekt platziert hat. Nun entscheidet der Server ob die Bewegungen mit der Computermaus und das sonstige Vorgehen des Benutzers menschlich waren oder der Vorgehensweise eines Bots ähneln. Das Ergebnis dieser Entscheidung wird dem Benutzer jedoch erst nach dem Drücken einer weiteren Schaltfläche präsentiert [Are14c].

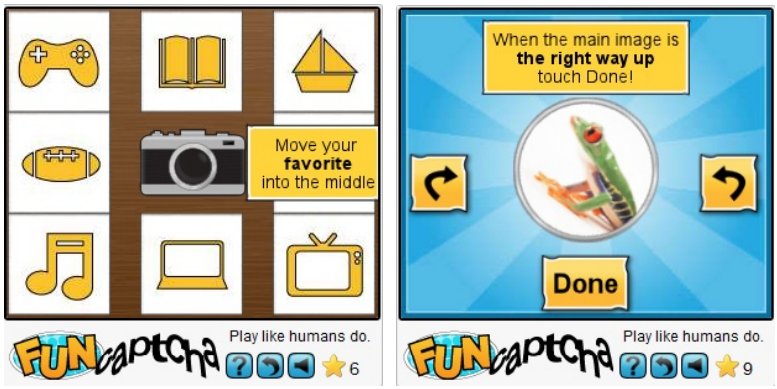
Abbildung 2.10: Ein PlayThru von Are You a Human [Are14c], bei dem der Spieler Objekte erkennen und platzieren muss. In diesem Beispiel müssen die Samen in das dafür vorgesehene Beet gestreut werden.

Diese Art von HIPs ist wesentlich benutzerfreundlicher als herkömmliche textbasierte HIPs. Zwar kann für das Lösen eines solchen PlayThrus durch die Ladevorgänge mehr Zeit benötigt werden als für das Lösen von textbasierten Aufgaben, jedoch ist das Frustrationslevel sehr viel geringer. Als Beispiel führt Are You a Human den Fall der Karriereseite HiredMyWay an [Are14a]. Die Benutzer der Seite beschwerten sich häufig über die lästigen und unbrauchbaren HIPs, die vor allem im Bereich der Registration von neuen Benutzern Verwendung fanden. Durch den Einsatz des HIPs von Are You a Human konnte nicht nur die allgemeine Benutzerzufriedenheit erhöht, sondern auch die Zeit, die für das Registrieren von neuen Kunden benötigt wurde um 2 Minuten gesenkt werden. Zudem füllten anstelle von 63% nun 99% der neuen Benutzer ihr Profil vollständig aus. Im Forum von VoteSane konnten unerwünschte Spam-Kommentare von über 10000 pro Monat durch die Verwendung von PlayThrus auf null gesenkt werden [Are14b].

Die PlayThrus können auf privaten Webseiten unentgeltlich verwendet werden. Für die kommerzielle Verwendung besteht außerdem die Möglichkeit, Werbung in den PlayThrus zu platzieren und die Aufgabenstellung in verschiedenen Sprachen anzuzeigen. In der kostenlosen Version ist die Anleitung stets in Englisch verfasst, was sich negativ auf die Gebrauchstauglichkeit auswirken kann.

Ein weiteres spielbasiertes HIP wird von FunCAPTCHA angeboten [Fun]. Bei diesem HIP muss der Benutzer zwei kurze Spiele korrekt lösen. Dabei muss entweder ein Objekt gefunden und an eine vorgegebene Stelle gezogen werden oder ein Bild in die korrekte Ausrichtung gedreht werden (siehe Abbildung 2.11). Je nach benötigter Zeit können mehr oder weniger Punkte gesammelt werden. Derzeit können diese Punkte nur gesammelt werden, die Entwickler erklären jedoch, dass diese später auch in irgendeiner Weise eingelöst werden können. Inwieweit dies einem *third party* Angriff (siehe Abschnitt 3.1) zugutekommt, sollte überdacht werden. Andernfalls würde möglicherweise ein Anbieter, der HIPs gegen Bezahlung löst, dadurch zusätzlich belohnt werden. Die Vergabe dieser Punkte stellt wiederrum eine Art von Gamifizierung dar. Zwar wird dies im Kontext eines Spiels eingesetzt, jedoch können die Benutzer durch das Sammeln der Punkte noch weiter motiviert werden.

Auch bei diesem HIP entstehen sehr lange Ladezeiten, die den Benutzer frustrieren könnten. Ebenfalls ist die Aufgabenstellung nur in Englisch verfügbar. FunCAPTCHA bietet jedoch auch an, das HIP an die Bedürfnisse des Seitenbetreibers anzupassen. Dazu gehören Farbthemen, Logos und personalisierte Willkommensnachrichten [Fun].



(a) In diesem Spiel müssen Objekte gefunden und in der Mitte abgelegt werden.

(b) In diesem Spiel muss ein Bild in die korrekte Ausrichtung gedreht werden.

Abbildung 2.11: Zwei unterschiedliche Spiele des FunCAPTCHA HIPs [Fun], die nacheinander gelöst werden müssen.

2.5 Grundlegende Methodik

Es gibt zwei verschiedene Möglichkeiten ein HIP System zu realisieren. Für textbasierte HIPs bietet sich folgende Lösung an. Mit dem Besuch der Seite wird ein HIP generiert. Dazu wird eine zufällige Folge von Zeichen generiert, die anschließend auf verschiedene Arten unleserlich gemacht wird. Um die Eingabe des Benutzers später überprüfen zu können, muss das System die Lösung sicher abspeichern. Um serverseitig Speicherplatz zu sparen, kann diese Lösung als Cookie zusammen mit dem generierten HIP an den Client weitergegeben werden. Dazu wird aus der Lösung zusammen mit einem geheimen Schlüssel ein Hash-Wert generiert [PK05]. Dadurch kann ein Angreifer zwar die Lösung besitzen, diese jedoch nicht lesen. Gibt ein Benutzer nun einen Text ein, wird dieser zusammen mit der verschlüsselten Lösung zum Server gesendet. Da dem Server der geheime Schlüssel bekannt ist, kann er aus dem eingegebenen Text wiederum einen Hash-Wert errechnen. Stimmen die beiden Hash-Werte überein, so handelt es sich um die korrekte Lösung und der Benutzer erhält Zugang [PK05].

Für HIPs deren Lösung komplexer ausfällt als einfacher Text, bietet es sich an, diese serverseitig zu speichern. Mit dem Besuch der Seite wird so zusätzlich zum HIP eine Session generiert. Die Lösung des HIPs wird in der Session gespeichert und verbleibt somit beim Server. Die ID der Session wird an den Client übertragen, der ab diesem Zeitpunkt dadurch identifiziert werden kann. Wenn der Benutzer das HIP löst, wird die Lösung zusammen mit der Session ID an den Server übertragen. Dieser lädt daraufhin die Session und kann nun die Lösungen miteinander vergleichen. Für jeden Benutzer wird so eine eigene Session erstellt, in der jeweils nur eine Lösung gespeichert wird. Wird dieser Ansatz nicht sauber implementiert, gibt es verschiedene Möglichkeiten dies für einen Angriff zu verwenden, was in Abschnitt 3.1 näher ausgeführt wird [Kal12].

2.6 Alternativen

Wie bereits in Kapitel 1 beschrieben, gibt es einige Alternativen die anstelle eines HIPs eingesetzt werden können. Da diese jedoch nicht im Fokus dieser Arbeit liegen, erfolgt an dieser Stelle nur eine kurze Beschreibung. Zudem werden nur einige der Alternativen behandelt, die sich eignen um ein Formular gegen automatisierte Angriffe zu schützen, da die Methoden für andere Zwecke zu speziell sind und für den jeweiligen Fall angepasst werden müssen.

- **Honeypot:** Hierbei handelt es sich um ein nicht sichtbares Eingabefeld, das jedoch im Quelltext der Internetseite zu finden ist. Da ein Bot nur mit diesem Quelltext arbeitet, besteht die Möglichkeit, dass er dieses Feld ebenfalls ausfüllt. Der Server kann den Inhalt dieses Feldes überprüfen und so einen Bot erkennen [Mot13, Tul].

- **Checkbox:** Die Checkbox funktioniert ähnlich wie der Honeypot, ist jedoch sichtbar. Die Beschriftung der Checkbox teilt dem Benutzer mit, dass er an dieser Stelle keinen Haken setzen soll. Auch hier besteht die Möglichkeit, dass ein Bot die Beschriftung nicht auswertet und den Haken setzt [Mot13].

- **Zeitstempel:** Da Bots im Gegensatz zu Benutzern ein Formular unter Umständen sehr schnell ausfüllen, können sie so entdeckt werden. Dazu wird im Formular ein Zeitstempel gespeichert, der nach dem Absenden ausgewertet wird. Ist die vergangene Zeit zu gering, handelt es sich um einen Bot [Mot13, Tul].

- **Dynamische Feldnamen:** Da Bots eine vorgefertigte Anfrage immer wieder an den Server senden können, werden hier die Namen der Eingabefelder variiert, und eine ungültige Anfrage kann erkannt werden [Tul].

- **Eingabefilter:** Bots versenden zumeist Nachrichten, welche Links enthalten. Je nach Kontext kann dies für einen Benutzer untypisch sein, wie beispielsweise in einem Kontaktformular. Durch die Überprüfung des Inhaltes der Anfrage kann ein Bot möglicherweise erkannt werden.

Bei dieser Liste handelt es sich nur um einen Auszug der am häufigsten verwendeten Techniken. Investiert der Angreifer genug Zeit, kann er die meisten dieser Mechanismen umgehen. Beispielsweise können die Zeitstempel der Formulare entsprechend geändert werden, sodass diese immer gültig sind. Und vergleicht ein Angreifer die Daten des Bots mit der Internetseite, wird er den Honeypot auch in kurzer Zeit identifiziert haben. Es handelt sich hierbei jedoch um leicht zu implementierende Abwehrmechanismen, die es einem Angreifer zumindest etwas schwerer machen, während normale Benutzer davon überhaupt nicht betroffen sind.

Sofern möglich sollte ein Angreifer nicht überprüfen können ob er erfolgreich war [HCR09]. Im Fall eines Kontaktformulars ist dies besonders anschaulich: Möchte ein Angreifer einen Seitenbetreiber mit Spam belästigen, wird er dessen Kontaktformular immer wieder absenden. Wenn die zuvor aufgeführten Abwehrtechniken greifen, sollte dies dem Angreifer verborgen bleiben. Dadurch geht der Angreifer davon aus, dass sein Angriff erfolgreich ist. Denn ihm ist es nicht möglich zu überprüfen ob tatsächlich eine E-Mail an den Seitenbetreiber versendet wurde oder nicht.

2.7 Zusammenfassung

Textbasierte HIPs werden mit Abstand am häufigsten eingesetzt [YSEA08b, Sut12]. Dies mag aus Gründen der Sicherheit gerechtfertigt sein, kann aber ebenso gut der Grund dafür sein, dass HIPs im Allgemeinen unbeliebt sind. Denn die Lesbarkeit dieser HIPs ist nicht in allen Fällen gegeben [HGH08].

Audiobasierte HIPs werden oft als Alternative zu textbasierten HIPs angeboten. Eigentlich sollte mit dieser Maßnahme blinden Benutzern der Zugang ermöglicht werden. Verschiedene Untersuchungen zeigten jedoch, dass Blinde die Audiosamples nicht wesentlich besser verstehen als Sehende und zudem durch schlechtes Design der Interfaces bei der Eingabe der Lösung zusätzlich behindert werden [BAB+08, BC09, BBP+11, RC13].

Bildbasierte Ansätze sind weniger häufig anzutreffen, bieten dann aber eine willkommene Abwechslung. Auch hier stellt es eine Herausforderung dar, eine ausreichend große Menge an Aufgaben bereit zu halten, da herkömmliche bildbasierte Ansätze, die Fotografien verwenden, die Aufgaben nicht automatisiert generieren können [EDHS07, Gol08]. Neuere Ansätze generieren Bilder oft aus verschiedenen Formen, bei der eine vorgegebene Eigenschaft gefunden und angeklickt werden muss [Lin]. Die Aufgabenstellung ist hierbei jedoch meist in Englisch, was nicht alle Internetnutzer verstehen.

Videobasierte HIPs haben sich nicht durchgesetzt. Dies könnte daran liegen, dass bei der Beschreibung des gesehenen Ausschnitts zu viel Spielraum für Interpretationen bleibt [JS12]. Auch ist es schwer eine angemessene Menge von Videos mit den entsprechenden Lösungen bereit zu halten. Da diese nicht automatisch erstellt werden können, muss dies von den Entwicklern des jeweiligen HIPs erledigt werden. Auch scheint die Verwendung dieser HIPs im mobilen Bereich ein zu großes Datenvolumen zu fordern.

Durch den Einsatz von spielbasierten HIPs kann die Benutzerzufriedenheit gesteigert werden [Are14a]. Leider sind diese HIPs noch nicht weit verbreitet und das Lösen dieser Aufgaben dauert inklusive der Ladezeiten meist länger als das Entziffern von textbasierten HIPs. Zudem sind auch hier die Anleitungen zu den wechselnden Aufgaben meist in Englisch verfasst [Are14c, Fun]. Jedoch können diese HIPs aufgrund ihrer erhöhten Komplexität sicherer gegen Angriffe sein.

Noch besser als spielbasierte HIPs ist der Einsatz gar keines HIPs [Bri11]. Wie bereits in Kapitel 1 beschrieben, existieren diverse Möglichkeiten Formulare gegen automatisierte Angriffe zu schützen [Mot13, Tul]. Da diese Maßnahmen jedoch bei einer sehr genauen Untersuchung des Formulars entdeckt werden können, sollte die Entwicklung von HIPs nicht gestoppt werden.

3 Angriffe auf HIPs

In diesem Kapitel wird eine Auswahl von wissenschaftlich durchgeführten Angriffen auf verschiedene HIPs erläutert. Dadurch soll greifbar werden, wie diffizil die Entwicklung eines neuen HIPs ist, und welche Aspekte es dabei zu berücksichtigen gilt. Die Mehrzahl der Angriffe erfolgte auf textbasierte HIPs, was damit in Zusammenhang steht, dass diese Art auch die meistverbreitetste ist. Dem folgen Angriffe auf bild-, audio- und spielbasierte HIPs.

Neben der Kategorisierung in diese vier Arten, ist die Auflistung alphabetisch nach dem Namen des HIPs oder dessen Anbieter sortiert. Wurden zu einem HIP mehrere Angriffe veröffentlicht, so sind diese chronologisch sortiert, um die Weiterentwicklung des HIPs aufzuzeigen.

Die Entwicklung von HIPs ist ein Wettlauf gegen die Angreifer, die immer ausgefeiltere Techniken entwickeln. Dabei müssen die Angreifer schnell und effektiv sein, da sie sonst Gefahr laufen entdeckt zu werden [Pon06]. Würden sie entdeckt werden, würden möglicherweise die HIPs angepasst und der Angriff müsste von Neuem entwickelt werden. Um die Angreifer dennoch aufzuspüren, müssen HIP-Systeme nicht nur richtige und falsche Antworten speichern, sondern auch wie viele Tests überhaupt von den Benutzern angefordert wurden. Sonst kann der Angreifer sein System trainieren ohne Konsequenzen fürchten zu müssen [Pon06].

Ein HIP gilt als gebrochen, wenn ein Angriff in 0,01 Prozent der Fälle eine korrekte Lösung liefern kann, die Erfolgsquote für den menschlichen Benutzer sollte dagegen bei etwa 80 Prozent liegen [CS05]. Die meisten hier im weiteren Verlauf beschriebenen Angriffe sind jedoch weitaus erfolgreicher. Bei manchen HIPs sind die durchgeführten Angriffe sogar erfolgreicher als ein menschlicher Benutzer.

Bei der Entwicklung von HIPs sollte eine ähnliche Vorgehensweise wie in anderen Bereichen der IT-Sicherheit gewählt werden [YSEA07]. So sollte die Sicherheit eines HIPs auf der verwendeten Methodik beruhen, und nicht auf deren Geheimhaltung. Leider wurde letzteres bei vielen HIPs versucht, wird dann jedoch die dahinter wirkende Technik bekannt, so ist das HIP schnell gebrochen. Denn wie schwierig es sein kann, ein gebrochenes HIP durch Anpassungen wieder abzusichern, wird vor allem durch die Fälle der HIPs von Microsoft (siehe Abschnitt 3.2.5) und Yahoo (siehe Abschnitt 3.2.8) erkennbar.

In den meisten Fällen wurden für die Entwicklung und den Test der Angriffe jeweils 100 Exemplare des jeweiligen HIPs verwendet. Sollte ein Angriff anhand weniger Exemplare getestet worden sein, so wird dies an entsprechender Stelle angegeben. Gleiches gilt, sollten die Verfasser des Angriffes keine Angaben hierzu gemacht haben. Wird eine Form maschinellen Lernens oder ein Satz von Vorlagen verwendet, ist es darüber hinaus wichtig, dass hierfür nicht mehr Exemplare als für den Test verwendet wurden. Nur so ist sichergestellt, dass es sich wirklich um einen effektiven Angriff handelt. Denn sonst muss möglicherweise zu viel Vorarbeit geleistet werden, um ein HIP effektiv angreifen zu können. Und diese Vorarbeit könnte von den Betreibern des HIPs entdeckt und der Angriff vereitelt werden.

3.1 Allgemeine Angriffe

In den Anfangsjahren wiesen die HIPs große Schwachstellen auf, welche durch einfachste Mittel ausgenutzt werden konnten. Da derartige Schwachstellen bei allen Arten von HIPs auftreten können, werden diese in diesem Abschnitt dargelegt. Grundsätzlich sollte darauf verzichtet werden, Parameter bezüglich des HIPs auf der Clientseite zu speichern und von dort via GET oder POST Anfrage[1] zum Server zu übertragen. Ein sehr gutes Beispiel hierfür wurde von Marcussen [Mar11] veröffentlicht. Dabei wurde der Text als GET Anfrage an den Webserver übertragen: In folgender URL stellt *aZcG5x* den Text dar, der vom Server in das HIP eingesetzt wird, welches dann wiederrum vom Benutzer gelöst werden muss: `http://domain.com/captcha/image.php?w=aZcG5x`.

Ein Hacker kann sich dies zunutze machen, da er den Lösungstext einfach aus der GET Anfrage extrahieren kann, oder einen eigenen Text an dieser Stelle einsetzt. Gleiches gilt für die Übertragung per POST Anfrage, da diese ebenfalls manipuliert werden könnte. Alle Entscheidungen, wie die Auswahl des Textes, Anzahl der Zeichen oder Auswahl des Spiels sollten ausschließlich auf der Seite des Servers, und somit außerhalb der Reichweite des Angreifers, erfolgen. Oftmals werden die Lösungen als Hash-Wert übertragen (siehe Abschnitt 2.5) [PK05]. Ist die Anzahl der möglichen Antworten gering, können die Hash-Werte für alle Antworten gespeichert und den Aufgaben zugeordnet werden [HCR09]. Das HIP muss in diesem Fall gar nicht mehr gelöst werden.

Wie bereits im Abschnitt 2.5 beschrieben, wird für das serverseitige Speichern der Lösung des HIPs meist eine HTTP Session verwendet. Erhält der Benutzer durch korrektes Lösen des HIPs Zugang zur Webanwendung, so gibt es folgende Möglichkeiten. Wird an dieser Stelle dieselbe Session weiterverwendet, kann entweder eine neue Session ID vergeben werden, oder diese wird ebenfalls beibehalten. Bleibt es bei derselben Session ID, kann dieselbe Lösung des HIPs zusammen mit der ID erneut zum Server gesendet werden [Kal12]. Da es sich um eine gültige Session ID und die zugehörige Lösung des HIPs handelt, wird die gewünschte Operation immer wieder durchgeführt. Wird eine neue Session ID vergeben, kann die alte Anfrage wie zuvor wiederverwendet werden, es muss jedoch die alte Session ID durch die neue ersetzt werden [Kal12]. Diese Schwachstelle kann dadurch behoben werden, dass entweder eine neue Session verwendet, oder die gespeicherte Lösung aus der Session entfernt wird [Kal12, Pur05].

Eine weitere Angriffsmöglichkeit besteht darin, die HIPs von Menschen lösen zu lassen. Dieser Angriff wird *third party attack* genannt und soll an einem Beispiel erläutert werden. Im Falle dass dieser Angriff für das Erstellen von E-Mail Konten verwendet wird, wird das Antragsformular per Skript ausgefüllt während das zu beantwortende HIP von Menschen gelöst wird [WJL12]. Diese tun das entweder gegen Bezahlung oder aufgrund eines anderen Anreizes. In Indien existieren Anbieter, die ihren Beschäftigten zwei US-Dollar pro 1 000 korrekt gelöster HIPs bezahlen [WJL12]. Da Hacker und Spammer jedoch immer auf der Suche nach kostengünstigeren Möglichkeiten sind, gibt es noch eine andere Vorgehensweise. Dabei werden Internetseiten mit meist pornographischen Inhalten erstellt, deren Zugang durch ein HIP gesichert ist [Kot04]. Hierbei handelt es sich allerdings um ein HIP des Mailproviders, bei dem der Hacker neue Benutzerkonten erstellen möchte. Löst ein Besucher der Pornoseite das HIP, so wird diese Antwort zusammen mit den übrigen Formulardaten, die zum Erstellen des E-Mailkontos benötigt werden, an den Mailprovider gesendet. Im Falle dessen, dass der Besucher der Pornoseite das HIP korrekt löste, konnte der Hacker so ein weiteres E-Mailkonto registrieren [GWLY11].

[1] Bei GET und POST handelt es sich um die zwei Möglichkeiten Parameter im Rahmen des HTTP zu übertragen. Die Parameter können anschließend vom Webserver ausgewertet werden (http://de.wikipedia.org/wiki/Hypertext_Transfer_Protocol#Argument.C3.BCbertragung).

Dieses Prinzip wurde oft kopiert und mit leichten Abwandlungen eingesetzt. Im Jahr 2007 war eine Malware im Umlauf, die einzelne Bilder eines Striptease zeigte. Um ein weiteres Bild zu sehen, musste der Benutzer des infizierten Computers ein weiteres HIP beantworten. Die eingegebene Lösung des HIPs wurde wie zuvor für andere Zwecke verwendet [GWLY11].

Gegen diesen Angriff gibt es derzeit keinen wirksamen Schutz, da schließlich ein Mensch das HIP beantwortet. Es könnte versucht werden, die IP Adresse dieser indischen Dienstanbieter oder manipulierten Pornoseiten in einer Blacklist zu verwalten und zu sperren. Doch sind die Möglichkeiten diese Sperre zu umgehen mannigfaltig, was an dieser Stelle jedoch nicht weiter behandelt werden soll, da dies nicht im Fokus dieser Arbeit liegt.

Die Schwachstellen bezüglich der GET oder POST Anfragen wurden im Laufe der Entwicklung immer seltener und sind heute nicht mehr zu finden [HCR09]. Eine Veröffentlichung aus dem Jahr 2012 zeigt, dass die Schwachstelle bezüglich der Session IDs vereinzelt noch immer besteht [Kal12]. Die in diesem Kapitel folgenden Angriffe beschäftigen sich jedoch nicht damit die HIPs zu umgehen, sondern versuchen das HIP so zu lösen wie es auch ein menschlicher Benutzer tun würde.

3.2 Angriffe auf textbasierte HIPs

Chellapilla et al. [CLSC05b] zeigten, dass Computer einzelne Zeichen mit größerer Sicherheit korrekt identifizieren können, als Menschen dazu in der Lage sind. Dieser Vorsprung wird umso größer, desto undeutlicher die Zeichen werden. Wird also ein einzelnes Zeichen deformiert und verzerrt dargestellt, kann es der Computer mit größerer Wahrscheinlichkeit als der Mensch korrekt identifizieren. Das bedeutet, dass textbasierte HIPs als gebrochen betrachtet werden können, sobald es möglich ist, die einzelnen Zeichen zu segmentieren und aus dem Bild zu extrahieren [CLSC05b, YSEA08a].

Die heute verwendeten textbasierten HIPs beruhen auf dieser Tatsache und versuchen durch Rauschen in Form von Linien und Bögen und durch die Überlappung der Zeichen die Segmentierung zu verhindern [SEAYM10]. Aus diesem Grund gleichen sich die einzelnen Schritte der Angriffe im Allgemeinen: Entfernen des Rauschens und eventuell verwendeter Farbe durch die Konvertierung in Schwarz-Weiß Bilder durch einen Schwellwertfilter. Anschließend erfolgt die Erkennung der Vordergrundpixel, also an welcher Position und Ausdehnung sich überhaupt Zeichen befinden. Dem folgt die Segmentierung des Textes und anschließend die Identifikation der extrahierten Zeichen [HGH08]. Die einzelnen Schritte sind dabei je nach HIP unterschiedlich intensiv oder können sogar entfallen. In den folgenden Abschnitten werden jedoch auch frühere HIPs beschrieben, bei denen die Angriffe bereits mit einfachen OCR Prozessen realisiert wurden.

Unter OCR versteht man das Identifizieren von Zeichen aus einer digitalen Bildvorlage. So kann aus einem gescannten Dokument, das als Bild vorliegt, mittels OCR eine Textdatei erstellt werden. Dabei existieren drei grundlegende Techniken, wie OCR durchgeführt werden kann [CL96]:

- **Dissection:** Hierbei handelt es sich um den üblichen Ansatz. Das Bild wird in einzelne Buchstaben zerlegt. Dafür werden die Eigenschaften der Buchstaben, wie Breite, Höhe und die Position genutzt. Der primitivste Ansatz unterteilt hierbei an den weißen Stellen des Textes. Dies ist sehr einfach bei gleicher Buchstabenbreite, kann jedoch auch mit vertikaler Projektion erreicht werden. Bei der vertikalen Projektion werden die Pixel des Bildes spaltenweise auf eine Ebene projiziert. So entsteht ein Histogramm, dessen Werte die Anzahl farbiger Pixel in einer Spalte angeben. An den Stellen, an denen sich im Histogramm keine oder nur sehr wenige Pixel befinden, kann der

Text aufgetrennt werden. Diese Technik wird von vielen der folgenden Angriffe eingesetzt. Ein anderer Ansatz wird *connected component* genannt. Dabei wird ein farbiger Pixel ausgewählt und die gleichfarbigen Pixel in der Nachbarschaft hinzugefügt. Diese Pixel werden wiederum einer solchen Prüfung unterzogen, bis keine angrenzenden gleichfarbigen Pixel mehr gefunden werden können. Durch diesen Vorgang kann ein komplettes Textzeichen gefunden werden [CL96]. Diese Methode wird im Folgenden in etwas abgewandelter Form als *color filling* bezeichnet.

- **Classification-based segmentation:** Das Bild wird von einem beliebig breiten Fenster durchfahren. Der aktuelle Bildausschnitt wird dabei mit gelernten Vorlagen verglichen. An der Stelle, an welcher der Klassifizierer den höchsten Wahrscheinlichkeitswert für einen Treffer liefert, wird segmentiert. Hierbei kann es helfen, die bisher erkannten Buchstaben zu Rate zu ziehen. So folgt im Englischen auf ein „th“ eher ein „e“ als ein „c“ [CL96]. Da viele der textbasierten HIPs zufällige Zeichenfolgen verwenden, kann diese Technik nur sehr selten eingesetzt werden.

- **Holisitc:** Das Wort wird als Ganzes erkannt und so segmentiert. Dafür wird jedoch ein sehr umfangreiches Wörterbuch benötigt [CL96]. Dieser Ansatz kann ebenfalls nur bei HIPs anwendet werden, die bekannte Wörter verwenden.

Beim *color filling* wird versucht, das Bild in einzelne Objekte zu zerlegen. Dabei werden diese Objekte mit unterschiedlichen Farben versehen, was eine spätere Identifikation erleichtert. Hierfür wird zunächst ein Pixel in Textfarbe gesucht. Dessen benachbarte Pixel werden nun ebenfalls auf ihre Farbe hin überprüft und gegebenenfalls gruppiert. Dies wird fortgeführt bis keine Nachbarpixel mehr mit gleicher Farbe gefunden werden können. Diese Gruppe von Pixeln wird dann mit einer anderen Farbe eingefärbt um diese nicht nochmals bearbeiten zu müssen. So kann das komplette Bild abgesucht und die Objekte gezählt werden. Bei HIPs mit einer festen Anzahl von Zeichen kann so beispielsweise geklärt werden ob sich noch andere Objekte, wie beispielsweise Rauschen, im Bild befinden. Die Objekte können nun weiteren Untersuchungen hinsichtlich der Breite, Höhe oder Anzahl der Pixel unterzogen werden. Dieses Verfahren gelingt jedoch nur, wenn die Zeichen nicht mehr verbunden sind, sich also Lücken zwischen ihnen befinden. Andernfalls würde möglicherweise nur ein einziges sehr großes Objekt gefunden werden.

Huang et al. [HLBO10] klassifizierten das Rauschen, welches verwendet wird um eine Segmentierung des Bildes zu verhindern. Dabei wurden 9 verschiedenen Arten von Rauschen identifiziert und sollen an dieser Stelle kurz beschrieben werden, da sie einen ersten Einblick in diese Methodik geben.

- Elemente mit Hintergrundfarbe: Diese schneiden meist die verwendeten Zeichen und hinterlassen so Bruchstücke, die eine Segmentierung stören können.

- Nicht schneidende Elemente kleiner Größe: Stellen für Segmentierungsalgorithmen kein großes Problem dar, sie können zum Beispiel durch das Zählen der zusammenhängenden Pixel erkannt werden.

- Nicht schneidende Elemente mit gerader Form: Diese Elemente besitzen meist die Größe von Buchstaben und können deshalb damit verwechselt werden.

- Nicht schneidende Elemente mit kurviger Form: Wie vorherige, ähneln jedoch zusätzlich in ihrer Form den verwendeten Buchstaben.

- Schneidende Elemente mit kleiner Größe: Stören im Allgemeinen wenig.

- Schneidende Elemente, die lang und dünn sind: Bei sehr großer Länge kann es passieren, dass mehrere Buchstaben miteinander verbunden werden. Dies erschwert die Segmentierung. Sehr dünne Elemente können jedoch in einem vorausgehenden Prozess entdeckt werden.

- Schneidende Elemente, die lang und dick sind: Befinden sich in den meisten Fällen ober- und unterhalb der Zeichen.

- Horizontal schneidende Elemente, die lang und dick sind: Flankieren meist Buchstaben und können so sehr oft mehrere Buchstaben miteinander verbinden und den Segmentierungsprozess stören.

- Vertikal schneidende Elemente, die lang und dick sind: Wie vorherige. Stellen jedoch keine wirkliche Schwierigkeit für die Segmentierung dar.

Zusätzlich zu den bisher erläuterten Techniken werden die Zeichen verzerrt. Dies allein steigert jedoch nicht die Sicherheit, weshalb es mit anderen Techniken kombiniert werden muss [CLSC05b]. Denn durch eine Verzerrung der Zeichen kann es erschwert werden, diese Zeichen von anderen Linien oder Objekten zu unterscheiden [HLBO10]. Huang et al. [HLBO10] konnten zusätzlich diese zwei möglichen Arten der Verzerrung identifizieren:

- Bei einer lokalen Anwendung wird das Zeichen an mehreren Stellen verzerrt. So ändert sich nicht die Form des Zeichens an sich, sondern es entstehen Wellen und das Zeichen erscheint mehr gekräuselt (siehe Abbildung 3.1(b)). Dies kann gegen Angriffe verwendet werden, die die Eigenschaften der Zeichen auswerten [HLBO10].

- Bei einer globalen Anwendung wird das ganze Zeichen verzerrt. So kann es beispielsweise gezogen und verbogen werden (siehe Abbildung 3.1(c)). Diese Methode erschwert Angriffe, bei denen die Zeichen gegen eine Vorlage verglichen werden [HLBO10].

(a) Buchstabe ohne Verzerrung. (b) Buchstabe mit lokaler Verzerrung. (c) Buchstabe mit globaler Verzerrung.

Abbildung 3.1: Verschiedene Arten von Verzerrung (aus [HLBO10]).

3.2.1 ASP Security Image Generator

Das HIP, welches der ASP Security Image Generator bis zum Jahr 2008 generierte, bestand aus 8 roten Ziffern vor einem weißen Hintergrund. Zudem wurde *pepper noise*[2] in ebenfalls roter Farbe eingesetzt, wie in Abbildung 3.2(a) zu sehen ist. Die Ziffern waren leicht verzerrt und zufällig in der Vertikalen verschoben [Klu08b].

[2] *Salt and Pepper noise* beschreibt das, häufig in Bildern vorkommende Rauschen aus weißen und schwarzen Pixeln. *Pepper noise* bezeichnet dementsprechend das Rauschen einfarbiger Pixel.

Kluever [Klu08b] knackte dieses HIP mittels maschinellen Lernens und setzte dabei verschiedene Techniken zur Erstellung des Trainingssets ein. Um die Bilder des HIPs segmentieren zu können, wendete Kluever im ersten Schritt einen Filter an, der das Rauschen entfernte. Dieser Filter berechnete schlicht die Durchschnittswerte für Pixelgruppen, so dass das Rauschen, welches nur aus jeweils einem Pixel bestand, abgeschwächt werden konnte (siehe Abbildung 3.2(b)). Zudem wurde dadurch das Bild in Graustufen umgewandelt und anschließend auf das Bild ein Schwellwertfilter angewendet, der das verschwommene Rauschen gänzlich entfernen konnte (siehe Abbildung 3.2(c)).

Die Segmentierung nahm Kluever [Klu08b] durch vertikale Projektion vor. Dabei werden die farbigen Pixel spaltenweise projiziert und aufsummiert. Spalten, die keine gefärbten Pixel enthalten, sind demnach Zwischenräume zwischen den einzelnen Ziffern, an denen das Bild unterteilt werden kann. Da die Ziffern leicht verschoben waren, wurden die Bilder, die nun nur noch die Ziffern zeigten, beschnitten und anschließend alle auf dieselbe Größe gebracht. Dies diente dazu, dass sich die Ziffern immer an derselben Position im Bild befanden (siehe Abbildung 3.2(d)). Dieses Vorgehen war bei jedem der getesteten Bilder erfolgreich [Klu08b].

(a) Beispiel eines HIPs des ASP Security Image Generator.

(b) Nach Anwendung des Filters.

(c) Nach Anwendung des Schwellwertfilters.

(d) Die segmentierten Ziffern.

Abbildung 3.2: Schritte zur Segmentierung des ASP Security Image Generator HIPs nach Kluever [Klu08b].

Für die anschließende Erkennung der Ziffern wurde die Korrelation zwischen den segmentierten Ziffern und Vorlagen errechnet, und die geringste Abweichung gewählt. Bei dem Angriff kamen drei unterschiedliche Vorlagensets zum Einsatz [Klu08b]:

- Da der Code des Generators verfügbar war, konnten Bilder mit vorgegebenen Ziffern generiert werden. Diese wurden auf oben genannte Weise behandelt und segmentiert. Dies bedeutet, dass diese Vorlagen ebenfalls Rauschen enthielten. Für jede Ziffer war genau eine Vorlage vorhanden. Mit dieser Technik konnten lediglich 4 von 100 getesteten HIPs korrekt gelöst werden.

- Kluever generierte und segmentierte die Bilder wie bereits zuvor, jedoch fand die Erzeugung der Bilder dieses Mal ohne Rauschen statt. Auch war für jede Ziffer wieder genau eine Vorlage vorhanden. Diese Vorgehensweise konnte 30 der 100 getesteten HIPs korrekt lösen.

- Kluever generierte erneut Bilder mit Rauschen und bearbeitete und segmentierte diese wie bereits beschrieben. Bei diesem Versuch wurden jedoch 800 Bilder pro Ziffer gespeichert und der Durchschnitt gebildet. Mit diesen Durchschnittsbildern als Vorlage konnten 72 von 100 getesteten HIPs korrekt gelöst werden.

Dass die Vorgehensweise des letzten Versuchs am erfolgreichsten war, spricht für die Vorgehensweise der allermeisten Angriffe, da in den wenigsten Fällen der Code tatsächlich verfügbar ist. So bleibt nur die Möglichkeit die Zeichen, die identifiziert werden müssen, aus dem HIP selbst zu extrahieren und zu klassifizieren. Jedoch ist das Vorlagenset mit 800 Bildern pro Zeichen sehr groß und müsste ohne

verfügbaren Quellcode des HIPs erst aufwändig gesammelt werden. Nach diesem Angriff wurde das HIP durch die Verwendung von Ziffern als auch Buchstaben und feinen Linien anstelle von *pepper noise* weiter verbessert [Klu08b].

3.2.2 Captchaservice.org

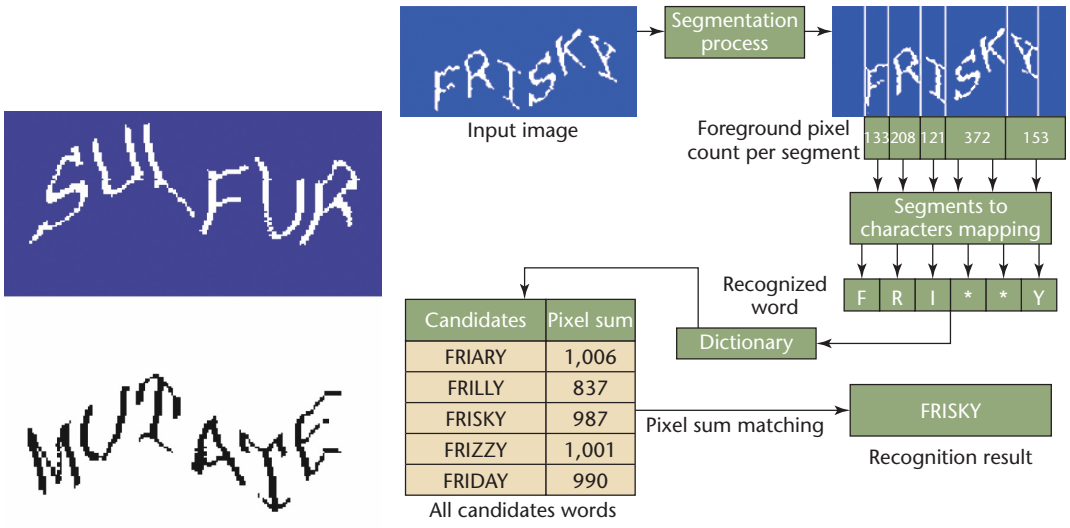
Der Dienst Captchaservice.org bot im Jahr 2007 vier unterschiedliche Modelle seiner HIPs an, die sich jedoch nur in Anzahl und Auswahl der Zeichen unterschieden: Ein Wort mit 6 Buchstaben, wobei ausschließlich englische Wörter Verwendung fanden, 6 zufällige Buchstaben, ein Wort, welches vom Anwender frei gewählt werden konnte und Ziffern unterschiedlicher Länge. Das frei gewählte Wort durfte nur Buchstaben in Groß- und Kleinschreibung sowie Ziffern enthalten. Das Modell welches nur Ziffern verwendete, tat dies aus Gründen der besseren Lesbarkeit ohne die Ziffern 5, 7 und 8 [YSEA07]. Bei allen Modellen waren die Zeichen durch eine Verschiebung der Pixel global verzerrt und leicht vertikal zueinander verschoben, wie Abbildung 3.3(a) zeigt. Zudem variierten die Farben des Vorder- und Hintergrunds [YSEA09].

Yan und Salah El Ahmad [YSEA07] konnten während ihrer Untersuchung einige Schwachstellen finden. So wurden zwar unterschiedliche, aber immer nur zwei Farben verwendet. Darüber hinaus kamen nur Großbuchstaben zum Einsatz, die aufgrund der speziellen Verzerrungstechnik immer die gleiche Pixelanzahl aufwiesen. Es kam hinzu, dass lediglich drei Buchstabenpaare dieselbe Pixelanzahl besaßen. Leider überschnitten sich die Zeichen nur in wenigen Fällen, so dass das Bild ohne größeren Aufwand segmentiert werden konnte. Das Modell mit zufälligen Ziffern variierte zusätzlich die Anzahl der Ziffern. Dabei kam es vor, dass nur eine Ziffer verwendet wurde. Ein solches HIP bietet keinen ausreichenden Schutz und kann bereits mit einfachen OCR Prozessen gelöst werden [YSEA07].

Zu Beginn versuchten Yan und Salah El Ahmad [YSEA07] verschiedene HIPs von Captchaservice.org mittels einer OCR-Software zu lösen. Dabei waren sie jedoch nur bei jenen erfolgreich, die aus nur einer Ziffer bestanden. Dies zeigt merkbar, dass das zufällige Verschieben von Pixeln eine gute Abwehr gegen OCR sein kann [YSEA09].

Im Anschluss daran wurden alle vier Modelle des HIPs einer Prüfung unterzogen. Dabei setzten die Verfasser zuerst vertikale Projektion ein um die Zeichen zu segmentieren, anschließend die Pixel der Zeichen zu zählen und mit den vorbereiteten Daten zu vergleichen. Wie bereits beschrieben, wiesen insgesamt nur 6 Buchstaben gleiche Pixelanzahlen auf. Aus diesem Grund konnten mit diesem einfachen Ansatz bereits 36 Prozent der HIPs, die ein Wort mit 6 Buchstaben enthielten, korrekt gelöst werden [YSEA07].

Eine Erweiterung dieses Angriffs schlug die gefundenen Buchstabenkombinationen in einem Wörterbuch nach und konnte so auf die verbleibenden Buchstaben schließen. Zudem sicherte diese Erweiterung gegen eine fehlgeschlagene Segmentierung ab, indem die Pixelzahlen der Wörter im Wörterbuch mit den gefunden Buchstaben und Pixelzahlen im Bild abgeglichen wurden. Abbildung 3.3(b) zeigt dies am Beispiel des Wortes *FRISKY*: Die Buchstaben S und K konnten nicht segmentiert werden, da hier eine Überschneidung vorlag. Jedoch konnten trotzdem die Pixel der Buchstaben gezählt und so auf das korrekte Wort geschlossen werden. Mit dieser Erweiterung konnten bereits 92% der HIPs korrekt gelöst werden. Durch die Verbesserung des Segmentierungsalgorithmus, bei dem nicht spaltenweise vorgegangen, sondern in Schlangenlinien der Raum zwischen den Buchstaben gesucht wurde, waren sogar 100% der HIPs korrekt gelöst worden. Pro HIP benötigte das Programm nur durchschnittlich 3,8 Sekunden [YSEA07].



(a) Zwei Beispiele eines HIPs von Captchaservice.org (aus [YSEA07]).

(b) Schematische Darstellung des Angriffs von Yan und Salah El Ahmad [YSEA07] anhand des Wortes FRISKY (aus [YSEA09]).

Abbildung 3.3: Beispiele des HIPs von Captchaservice.org und eine schematische Darstellung des Angriffs von Yan und Salah El Ahmad [YSEA07, YSEA09].

Für die HIPs mit 6 zufälligen Buchstaben konnte das Wörterbuch nicht verwendet werden, weshalb hierbei eine Analyse der Buchstabenform stattfand. So konnten auch 99% dieser HIPs in durchschnittlich fünf Sekunden korrekt gelöst werden. Die HIPs, welche ausschließlich Ziffern enthielten, konnten ohne Probleme segmentiert werden. Dadurch, dass bei diesen HIPs weitaus weniger unterschiedliche Zeichen verwendet wurden, gab es keine Übereinstimmung der Pixelzahlen. Dennoch konnten alle dieser HIPs in durchschnittlich 2 Sekunden gebrochen werden. Die HIPs, welche ein benutzerdefiniertes Wort verwendeten, wurden von Yan und Salah El Ahmad [YSEA07] nicht weiter untersucht. Jedoch merkten die Verfasser an, dass in diesem Fall diejenigen Buchstaben, die dieselbe Pixelzahl aufweisen, durch die Analyse ihrer Form identifiziert werden können. Es ist demnach davon auszugehen, dass Yan und Salah El Ahmad mit ihrem Angriff alle Modelle dieses HIPs erfolgreich brechen konnten.

3.2.3 Gimpy, EZ-Gimpy, Gimpy-r

Gimpy zeigte im Jahr 2003 zehn Wörter, von denen drei korrekt eingegeben werden mussten. Dabei wurden immer zwei Wörter übereinander geschrieben um die Lesbarkeit zu erschweren. Dieses Vorgehen liefert eine sehr hohe Sicherheit gegen Segmentierung, da einzelne Buchstaben nicht erkannt werden können. Zudem waren die Wörter vor einem verschiedenfarbigen Hintergrund gedruckt (siehe Abbildung 3.4(a)). Jedoch wurde beim generieren des HIPs nur aus 411 Wörtern zufällig ausgewählt [MM03].

EZ-Gimpy wurde im Auftrag von Yahoo entwickelt und verwendete nur ein Wort, das auch nur leicht verzerrt und nicht verschoben war [PK05]. EZ-Gimpy verzerrte das Wort zusammen mit dem farbigen Hintergrund global. Als Hintergrund kamen entweder Farbverläufe oder Muster, wie zum Beispiel Gitter zum Einsatz (siehe Abbildung 3.4(b)) [MM03]. Insgesamt konnten 561 unterschiedliche Wörter identifiziert werden [MM03]. Dieses HIP verwendete immer dieselbe Schriftart, was ebenfalls eine Schwachstelle darstellt [MJHP04].

Gimpy-r kann als Weiterentwicklung von EZ-Gimpy verstanden werden, da es einen viel höheren Grad von Verzerrung verwendete (siehe Abbildung 3.4(c)). Des Weiteren wurden zwei unterschiedliche Schriftarten und jeweils vier zufällige Buchstaben anstelle von Wörtern verwendet. Aus Gründen der besseren Lesbarkeit verwendete Gimpy-r jedoch nur 19 verschiedene Buchstaben [MJHP04].

(a) Gimpy (b) EZ-Gimpy (c) Gimpy-r

Abbildung 3.4: Beispiele für HIPs aus der Gimpy-Familie (aus [MM03]).

Da bei Gimpy die Wortpaare nicht exakt übereinander gedruckt wurden, konnten Wortanfänge und -enden meist besser gelesen werden. Diese Tatsache machten sich Mori und Malik [MM03] zu Nutze und grenzten die möglichen Wörter bereits dadurch stark ein, da nur etwa 100 unterschiedliche erste und letzte Buchstabenpaare vorhanden waren. Durch die Analyse der Buchstabenumrisse und das Abschätzen der Wortlänge konnten die in Frage kommenden Wörter auf durchschnittlich vier reduziert werden. Für jedes dieser Worte wurden die Pixel aus dem Bild entfernt und versucht, die verbleibenden Pixel dem zweiten Wort zuzuordnen. Daraus resultieren pro Wortpaar 16 mögliche Worte, die synthetisiert und mit dem Originalbild verglichen werden mussten. Die am besten passende Kombination wurde als Lösung ausgewählt und war in 33 Prozent der Fälle korrekt. Eine Anwendung dieses Vorgehens auf EZ-Gimpy resultierte in 92% korrekt gelösten HIPs [MM03]. Bei diesem Angriff handelt es sich um den ersten wissenschaftlich dokumentierten Angriff auf ein textbasiertes HIP [HCR09].

Auch Moy et al. [MJHP04] unterzogen EZ-Gimpy einer Prüfung und konnten 99 Prozent der getesteten HIPs lösen. Allerdings war ihr Angriff sehr komplex und dauerte pro HIP circa 18 Sekunden. Da unterschiedliche Hintergründe Verwendung fanden, wurde versucht die Farbe oder das Gitter durch die Suche von periodischen Linien zu entfernen. Zudem musste eventuell vorhandenes Rauschen unterdrückt werden, um anschließend das Bild in kleine Abschnitte zu unterteilen und mit Vorlagen zu vergleichen um auf die korrekten Buchstaben schließen zu können.

Gimpy-r HIPs konnten von Moy et al. [MJHP04] zu 78%, in jeweils circa 7 Sekunden, gebrochen werden. Nachdem der Hintergrund auf dieselbe Weise wie bei EZ-Gimpy entfernt wurde, sollte die Verzerrung der Buchstaben zurückgerechnet werden. Dazu zeichnete das entwickelte Programm Linien zwischen die einzelnen Buchstaben. Diese Linien befanden sich in gleichem Abstand zu den beiden jeweiligen Buchstaben. Nun wurde die Linie begradigt und die Buchstaben so erneut deformiert. Abbildung 3.5 zeigt die deformierten Linien zwischen den Zeichen, die noch begradigt werden müssen. Nun sollten die Buchstaben in etwa ihre ursprüngliche Form wieder angenommen haben. Diese Buchstaben konnten dann mit Vorlagen abgeglichen, und die Vorlage mit der kleinsten Abweichung ausgewählt werden.

Chellapilla und Simard [CS05] waren bei ihrer Untersuchung erstaunt, wie einfach es war, die von ihnen untersuchten HIPs zu segmentieren. Angeblich verwendeten sie keine ausgefeilten Algorithmen sondern durchsuchten das Bild lediglich nach schwarzen Pixeln, da die Schrift immer diese Farbe aufwies. Durch maschinelles Lernen mittels eines neuronalen Netzes konnten sie 34% der EZ-Gimpy HIPs korrekt lösen. Das Trainingsset hatte hierbei eine Größe von 1600 HIPs.

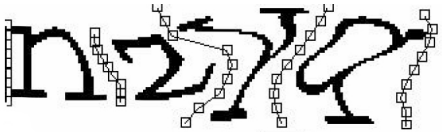


Abbildung 3.5: Gimpy-r HIP mit eingezeichneten Linien zum begradigen der Zeichen. Nach Vorlage von Moy et al. [MJHP04].

3.2.4 Google

Google setzte im Jahr 2005 ein HIP ein, um sich gegen den missbräuchlichen Einsatz seiner E-Mail Konten zu wehren. Dieses HIP generierte einfarbigen Text, dessen Farbe jedoch variierte, der vor einem weißen Hintergrund zu sehen war. Die Buchstaben waren sehr eng geschrieben, berührten sich aber nur unregelmäßig (siehe Abbildung 3.6(a)) [CS05].

Das Segmentieren war laut Chellapilla und Simard [CS05] ein leichtes, da keinerlei Abwehrmechanismen vorhanden waren. So konnten mittels maschinellem Lernen 5 Prozent der HIPs automatisiert beantwortet werden. Durch die Segmentierung durch *color filling* konnten Yan und Salah El Ahmad [YSEA08a] 12 Prozent der HIPs erfolgreich segmentieren. Die Identifikation der Buchstaben wurde im Rahmen ihrer Untersuchung nicht durchgeführt, die Verfasser rechneten allerdings mit einer gesamten Erfolgsquote von 9 Prozent. Google schloss diese Sicherheitslücke im Jahr 2008, die Buchstaben berühren sich nun immer, wie in Abbildung 3.6(b) zu sehen ist [YSEA08a]. Seitdem konnte dieses HIP nicht mehr gebrochen werden [GWF12].

(a) Google HIP aus dem Jahr 2005 (aus [CS05]).

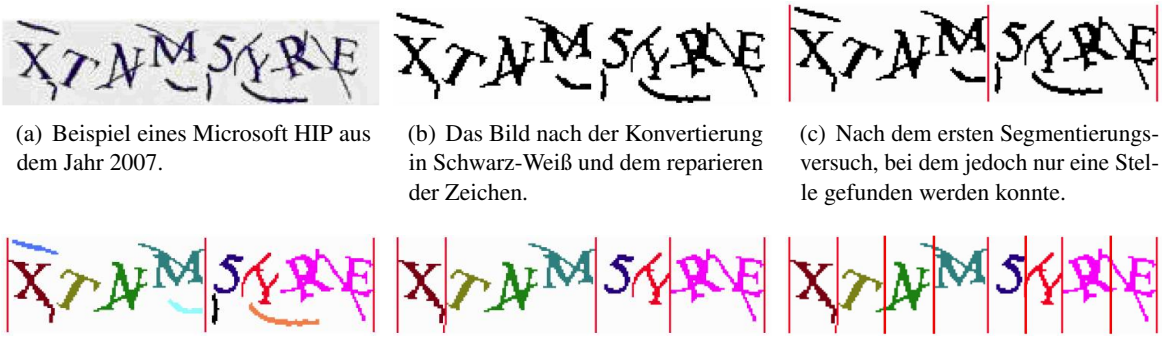
(b) Google HIP aus dem Jahr 2008 (aus [SEAYM10]).

Abbildung 3.6: Beispiele des Google HIPs vor und nach der Weiterentwicklung. In der neuen Version sind die Buchstaben viel enger gesetzt, was eine Segmentierung verhindern soll.

3.2.5 Microsoft

Das HIP von Microsoft wurde während seines Einsatzes immer weiter verbessert. Beispielsweise 2003 durch das Hinzufügen von Rauschen in Form von kleinen Linien, um gegen die Segmentierung vorzugehen [SSB+03]. Dieses Rauschen wurde 2005 noch weiter verbessert, und die Linien durch Bögen, die ähnliche Formen wie die Buchstaben aufweisen, ersetzt [CS05]. Im Jahr 2007 wies das HIP folgende Eigenschaften auf (siehe Abbildung 3.7(a)) [YSEA08a]. Es bestand aus 8 Großbuchstaben oder Ziffern, die zufällig ausgewählt wurden und kein Wort bildeten. Die Farben waren immer dieselben, ein grauer Hintergrund und dunkelblaue Zeichen. Die Verzerrung kam sowohl lokal als auch global zur Anwendung. Zudem wurden drei unterschiedliche Arten von Bögen eingesetzt: Dicke Bögen, die Teilen von Buchstaben ähnlich sahen, und um die Zeichen herum angeordnet waren. Dünnere Bögen, die Zeichen und dicke Bögen schneiden und dünne Bögen in Hintergrundfarbe, die ebenfalls Buchstaben schneiden und so einige Pixel entfernen.

Die Untersuchung von Yan und Salah El Ahmad [YSEA08a] bezieht sich auf das HIP, das Microsoft im Jahre 2007 eingesetzte. Das Ergebnis der Untersuchung wurde jedoch erst veröffentlicht, nachdem Microsoft die Sicherheitslücken behoben hatte. Um die HIPs vollautomatisch zu lösen, wandelte der erste Schritt das Bild in Schwarz-Weiß Farben um und versuchte die Buchstaben, die durch dünne Bögen geschnitten wurden zu reparieren (siehe Abbildung 3.7(b)). Dazu wurden Pixel gesucht, die von Vordergrundfarbe umgeben sind um dann ebenfalls so eingefärbt zu werden. Anschließend erfolgte ein erster Versuch, das Bild zu segmentieren. Wie in Abbildung 3.7(c) zu sehen, ist dies allerdings nur teilweise erfolgreich. Wenn das Bild nicht in acht Segmente unterteilt werden konnte, werden Objekte durch *color filling* Segmentierung gesucht (siehe Abbildung 3.7(d)). Nun wurde versucht, die dicken Bögen zu entfernen und nochmals zu segmentieren (siehe Abbildung 3.7(e)). Dicke Bögen konnten anhand ihrer relativen Position identifiziert werden, da sie immer um den Text verteilt waren und meist aus weniger Pixel bestanden als die Zeichen. Konnte das Bild an dieser Stelle nicht in acht Segmente unterteilt werden, musste abgeschätzt werden wie viele Zeichen die einzelnen Segmente enthalten. Dies konnte anhand der Segmentbreite in ausreichender Genauigkeit erfolgen. Die Segmente wurden anschließend in gleich große Teile aufgeteilt, wie Abbildung 3.7(f) zeigt [YSEA08a].



(a) Beispiel eines Microsoft HIP aus dem Jahr 2007.

(b) Das Bild nach der Konvertierung in Schwarz-Weiß und dem reparieren der Zeichen.

(c) Nach dem ersten Segmentierungsversuch, bei dem jedoch nur eine Stelle gefunden werden konnte.

(d) Objekte wurden mittels *color filling* gefunden und markiert.

(e) Die dicken Bögen wurden in diesem Schritt entfernt.

(f) Gleichmäßige Aufteilung der verbleibenden Segmente. Nun können die Zeichen identifiziert werden.

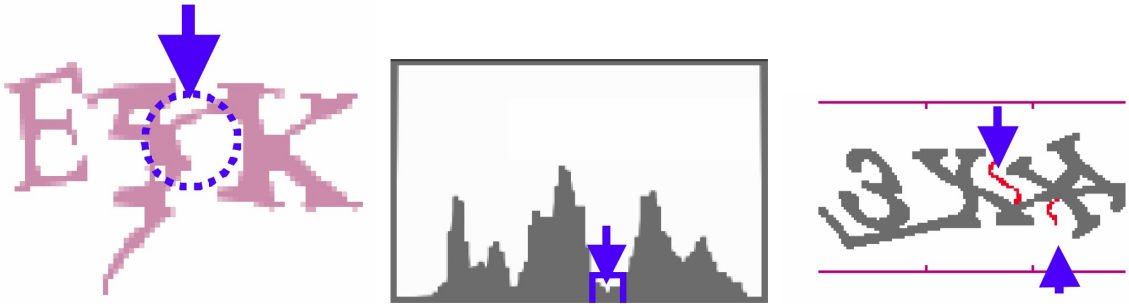
Abbildung 3.7: Einzelschritte der Segmentierung des Microsoft HIPs aus dem Jahr 2007 von Yan und Salah El Ahmad [YSEA08a].

Dieses Vorgehen führte in 91 Prozent der 500 getesteten HIPs zum Erfolg. Leider erfolgte hier keine Prüfung wie akkurat die Identifikation der Zeichen erfolgen kann. Yan und Salah El Ahmad [YSEA08a] gingen jedoch von einer 60 prozentigen Erfolgsquote aus. Eine Absicherung gegen diesen Angriff könnte durch die Überlappung der Buchstaben und durch die Anpassung der Bögen erfolgen. Dazu sollten die Bögen in ihrer Form und Position den Buchstaben gleichen. Ferner würden zufällige Buchstabenbreiten dagegen helfen, den Text in äquidistante Segmente unterteilen zu können [YSEA08a].

Huang et al. [HLBO10] geben an, für ihre Untersuchung HIPs aus dem Zeitraum Januar 2008 verwendet zu haben. Demnach handelt es sich um dieselbe Version des HIPs wie auch bei obiger Untersuchung. Dennoch sind die beiden Vorgehensweisen sehr unterschiedlich. Während Yan und Salah El Ahmad [YSEA08a] darauf verzichteten dünne Bögen zu entfernen, beseitigt dieser Angriff diese im ersten Schritt. Zudem wurde das Bild mehrmals erodiert und erweitert. Bei der Erosion werden die äußeren Pixel einer Form gelöscht, während beim Erweitern die Form durch eine Pixelreihe erweitert wird. So werden nach und nach feine Strukturen entfernt. Im nächsten Schritt erfolgte eine vertikale Projektion des Bildes. Das sich daraus ergebende Histogramm wurde bei diesem Ansatz von einem Fenster durchfahren. Bereiche, die auf der ganzen Breite des Fensters unterhalb des Schwellwerts (der Höhe des Fensters) lagen, wurden entfernt. Dabei war die Breite und Höhe des Fensters variabel und konnte dynamisch festgelegt werden. Ein großer Schwellwert entfernt zwar sicherer die gesuchten Linien, aber

auch Teile der Zeichen. Ein breiteres Fenster zerstört keine Zeichen, findet aber auch weniger Linien oder Bögen. Abbildung 3.8(a) zeigt eine Linie zwischen zwei Zeichen, die entfernt werden muss. Das Histogramm dieses Ausschnittes ist in Abbildung 3.8(b) gezeigt und wird von dem Fenster durchlaufen. Da die Projektion der Linie innerhalb des Fensters liegt, wird sie entfernt. Als idealen Schwellwert gaben Huang et al. [HLBO10] die Breite der Linien oder Bögen, und als optimale Breite des Fensters den Abstand der Zeichen an. Dieser Vorgang könnte bei einigen Zeichen zu Problemen führen, da diese in der Mitte sehr dünn sind (0, O, D, B und 8). Diese Zeichen wurden anhand ihrer geschlossenen Form identifiziert und von diesem Schritt ausgenommen [HLBO10].

Die anschließende Segmentierung nahmen Huang et al. [HLBO10] wie folgt vor. Die wahrscheinlichsten Stellen für eine Segmentierung wurden anhand der durchschnittlichen Buchstabenbreite festgelegt. An diesen Stellen wurde nach Bereichen gesucht, die von Pixeln mit Schriftfarbe umgeben waren. In der Mitte dieser Bereiche zog der Angriff Trennlinien, und zwar sowohl an der Unter- als auch an der Oberseite des Textes. Trafen sich zwei solche Trennlinien oder befanden sich an einer ähnlichen Position, so konnten diese verbunden, und der Text an dieser Stelle segmentiert werden [HLBO10]. Zwei Trennlinien an ähnlicher Position sind in Abbildung 3.8(c) gezeigt.



(a) Linie zwischen 5 und K, die entfernt werden soll.

(b) Histogramm der vertikalen Projektion mit dem durchlaufenden Fenster.

(c) Zwei Trennlinien die anschließend verbunden werden und die Segmentierung vornehmen.

Abbildung 3.8: Verschiedene Schritte der Segmentierung nach Huang et al. [HLBO10].

Daraufhin konnten die Objekte mittels *color filling* identifiziert werden. Befanden sich in einem Segment mehrere Objekte, so wurde das kleinere gelöscht, da es sich hierbei um eine nicht gelöschte Linie handeln musste. Auf diese Weise konnten 18 Prozent der getesteten HIPs gelöst werden [HLBO10].

3.2.6 Megaupload

Bei diesem HIP kamen nur die Farben Schwarz und Weiß zum Einsatz. Dabei bestand der Text hauptsächlich aus schwarzen Pixeln, während die Stellen der Zeichen, die sich überlappen, weiß dargestellt waren. Die Zeichen wurden rotiert und horizontal miteinander verbunden, so dass die Überlappungen einem festen Schema folgten (siehe Abbildung 3.9(a)). Zudem enthielt das HIP immer 3 Buchstaben gefolgt von einer Ziffer. Die Segmentierung dieses HIPs sollte dadurch verhindert werden, dass es keinen Zwischenraum zwischen den Zeichen gibt. Für eine bessere Lesbarkeit wurden die überlappenden Teile der Zeichen weiß gefärbt [SEAYM10]. Während dieses Vorgehen Programmen die Segmentierung erschweren sollte, haben Menschen mit dem Erkennen dieser so bearbeiteten Zeichen keine oder nur geringe Schwierigkeiten [CLSC05b]. Zugunsten der Gebrauchstauglichkeit wurden einige Zeichen (I, J, L, O und 0) nicht verwendet und nur Großbuchstaben eingesetzt. Zusätzlich waren alle Zeichen von derselben Schriftart, was die Sicherheit ebenfalls einschränkte [HCR09].

Anders als die anderen hier behandelten Angriffe beruht der von Salah El Ahmad et al. [SEAYM10] nicht auf Segmentierung, da dies hier aufgrund der fehlenden Zwischenräume nicht möglich war (siehe Abbildung 3.9(a)). Stattdessen wurden im ersten Schritt die schwarzen Flächen extrahiert. Da diese sich nur in den allerwenigsten Fällen direkt berührten, war dies besonders einfach, jedoch konnte ein Zeichen aus mehreren dieser Flächen bestehen (siehe Abbildung 3.9(b)). Um die Komplexität gering zu halten, wurden nun zu kleine Flächen entfernt und anschließend die weißen Flächen extrahiert. Leider sind weiße Flächen auch durch die Form der Zeichen gegeben. Im Abbildung 3.9(c) sind beispielsweise die Zwischenräume in A, Q und 6 gezeigt. Um diese zu identifizieren wurden die drei größten Flächen gesucht, und alle anderen Flächen in ihrer Umgebung entfernt. So blieben in den meisten Fällen nur die überlappenden Flächen übrig, wie Abbildung 3.9(d) zeigt. An dieser Stelle des Angriffs waren also nun alle Teile der Zeichen bekannt und mussten lediglich noch korrekt zusammengesetzt werden [SEAYM10].

Da die HIPs alle nach demselben Schema generiert wurden, war auch die Abfolge klar, in der die Teile wieder zusammengesetzt werden mussten: Das erste Zeichen besteht aus den jeweils ersten schwarzen und weißen Objekten. Das zweite Zeichen besteht aus den ersten beiden weißen und aus dem zweiten schwarzen Objekt usw. [SEAYM10]. Das Ergebnis ist in Abbildung 3.9(e) exemplarisch dargestellt.

(a) Beispiel eines Megaupload HIPs aus dem Jahr 2010.

(b) Die extrahierten kleinen und großen schwarzen Flächen.

(c) Hier sind alle gefundenen weißen Flächen markiert.

(d) Nach der Identifikation der korrekten weißen Flächen, konnten diese extrahiert werden.

(e) Die Zeichen, aus den schwarzen und weißen Flächen zusammengesetzt.

Abbildung 3.9: Verschiedene Schritte aus dem Verfahren von Salah El Ahmad et al. [SEAYM10].

Die Effizienz des Angriffs wurde anhand von 400 HIPs überprüft. 78 Prozent dieser HIPs konnten erfolgreich segmentiert werden. Da davon ausgegangen werden kann, dass segmentierte Zeichen zu 95 Prozent korrekt identifiziert werden können [CLSC05b], gehen Salah El Ahmad et al. von einer Gesamteffizienz von 64 Prozent aus. Für die Segmentierung der Zeichen benötigte dieser Angriff durchschnittlich nur 120 ms [SEAYM10].

Es konnten von Salah El Ahmad et al. [SEAYM10] folgende Verbesserungsmöglichkeiten aufdeckt werden: Die Breite sowie die Anzahl der Zeichen sollten variabel sein. Die Unterscheidung zwischen verdeckten Bereichen und Flächen innerhalb der Zeichen sollte erschwert werden und die Verbindung der Zeichen untereinander sollte zufälliger sein. Zudem könnte das Rotieren der Zeichen in zufälligem Winkel erprobt werden.

3.2.7 Paypal

Für sein HIP verwendete PayPal 2008 einen einfarbigen Hintergrund, graues Rauschen, sowie Gitternetze und schwarze Buchstaben. Die Zeichen der Schrift bestanden aus mehreren Teilen und wiesen Lücken auf, was möglicherweise eine Segmentierung verhindern sollte (siehe Abbildung 3.10(a)). Die Zeichen 1, I, O, Q und 0 fanden keine Anwendung, es wurden jedoch immer 5 Zeichen verwendet [Klu08a].

Ein großer Schwachpunkt dieses HIPs ist, dass für das Rauschen eine andere Farbe als für die Zeichen verwendet wurde. So kann das Rauschen mit einfachen pixelbasierten Techniken entfernt werden.

Der Angriff von Kluever [Klu08a] wirkt sehr einfach, trotzdem konnten damit alle getesteten HIPs gelöst werden. Und zwar wurden die Bilder zuerst in Graustufen umgewandelt um anschließend einen Schwellwertfilter anzuwenden. Durch diese einfache Technik konnten bereits die Zeichen extrahiert werden. Es mussten dann nur noch vereinzelte Pixel und Artefakte gelöscht werden. Das Ergebnis ist in Abbildung 3.10(b) gezeigt. Für die eigentliche Segmentierung wurde wiederrum vertikale Projektion eingesetzt. Aufgrund der festen Anzahl der Zeichen und der bekannten minimalen Zeichenbreite konnten die Lücken in der Schrift ignoriert werden [Klu08a].

(a) Ein Beispiel eines PayPal HIPs aus dem Jahr 2008.

(b) Aus dem PayPal HIP extrahierter Text, der anschließend erkannt werden kann.

Abbildung 3.10: Verschiedene Schritte des Angriffs von Kluever [Klu08a].

Kluever [Klu08a] testete mehrere Möglichkeiten die Zeichen zu identifizieren: Die Identifikation der Zeichen anhand ihrer Pixelanzahl gelang nur in 8 Prozent der Fälle. Die Berechnung der Korrelation der vertikalen Projektion lieferte hingegen 97% korrekt gelöste HIPs. Die Korrelation der horizontalen Projektion oder die Korrelation der Pixelpositionen konnte jedes der 100 HIPs richtig lösen. Für das Training wurden lediglich 20 Bilder verwendet [Klu08a].

3.2.8 Yahoo

Nachdem gezeigt wurde, dass die Gimpy HIPs nicht sicher waren, musste Yahoo 2005 ein neues HIP entwickeln. Dieses erhielt keinen Namen, weshalb es hier, anders als Gimpy, unter dem Namen Yahoo aufgeführt wird. Dieses HIP enthielt keine Wörter, sondern durchschnittlich insgesamt fünf zufällige Groß- und Kleinbuchstaben sowie Ziffern, und nur die Farben Schwarz und Weiß, wobei die Verteilung auf Hintergrund und Zeichen zufällig erfolgte. Zum Schutz vor Segmentierung kamen schwarze Linien zum Einsatz [CS05]. Zusätzlich enthielt das HIP einen schwarzen Rahmen und verzerrte die Zeichen global, rotierte sie teilweise und verschob sie in der Vertikalen (siehe Abbildung 3.11(a)).

Dieses HIP wurde im März 2008 durch eine neue Version mit variabler Zeichenanzahl abgelöst, welche die Zeichen auch gekippt darstellte. Die übrigen Eigenschaften wurden von der Vorgängerversion übernommen [YSEA08a]. Abbildung 3.11(b) zeigt ein Beispiel hierfür.

2012 verwendete Yahoo ein HIP, das segmentierungssicher sein sollte. Dazu wurde eine neue Technik verwendet, die Zeichen so eng zusammen schob, dass sich diese berühren. Weiterhin verwendete

dieses HIP sechs bis acht Groß- und Kleinbuchstaben sowie Ziffern. Der Hintergrund war nun weiß, die Zeichen dunkelgrau dargestellt. Zur besseren Lesbarkeit wurden nur bestimmte Buchstaben und Ziffern verwendet. Insgesamt fanden 10 Groß-, 12 Kleinbuchstaben und 7 Ziffern Verwendung, die eine globale Deformation erfuhren und sich in den meisten Fällen berührten. Zusätzlich waren die Zeichen von zerstreuten Punkten umgeben, die wie Artefakte einer starken Komprimierung wirken. Manche Texte wiesen eine sinusförmige Wellenform auf, andere waren flach. Bei wellenförmigen Texten wurde die Grundlinie der Zeichen verschoben, sodass sich die aufeinanderfolgenden Zeichen nicht zu weit voneinander unterschieden. Abbildung 3.11(c) zeigt dies besonders anschaulich.

(a) Das 2005 von Yahoo verwendete HIP (aus [CS05]).

(b) Das 2008 von Yahoo verwendete HIP (aus [YSEA08a]).

(c) Das 2012 von Yahoo verwendete HIP (aus [GWF12]).

Abbildung 3.11: Yahoo HIPs verschiedener Versionen.

Der Angriff von Chellapilla und Simard [CS05] wurde auf die Version von 2005 (siehe Abbildung 3.11(a)) angewendet und entfernte zuerst den Rahmen, die Linien durch erweitern und erodieren und wählte anschließend Pixelflächen aus, die in etwa die Größe von Zeichen hatten. Abbildung 3.12 zeigt das so bereinigte HIP. Nach eigenen Angaben konnten so 58 Prozent der HIPs segmentiert werden. Von den 900 getesteten HIPs konnten 46 Prozent korrekt erkannt werden. Bei diesem Ansatz wurden jedoch 1600 HIPs zum Trainieren des Klassifizierers eingesetzt.

Abbildung 3.12: Das Yahoo HIP aus dem Jahr 2005, durch den Angriff von Chellapilla und Simard [CS05] vorbereitet zur Segmentierung.

Yan und Salah El Ahmad [YSEA08a] wendeten ihren Angriff auf das Microsoft HIP (siehe Abschnitt 3.2.5) auch auf das Yahoo HIP, das bis März 2008 verwendet wurde, an. Bereits durch geringe Anpassungen konnten 77 Prozent der HIPs segmentiert werden. Dies bedeutet inklusive der Identifikation der Zeichen eine Gesamtquote von 60 Prozent [YSEA08a].

Das im März 2008 veröffentlichte HIP wies laut Yan und Salah El Ahmad [YSEA08a] einige grundlegende Fehler auf: Zwar wurde nun keine feste Zeichenzahl mehr verwendet, jedoch konnte die Anzahl durch Messen der Wortlänge abgeschätzt werden. Dies ermöglichte so eine einfache Unterteilung in Segmente. Die gekippten Zeichen konnten durch eine gekippte Segmentierung erfasst werden, wie in Abbildung 3.13 gezeigt wird. Dadurch könnten auch 26 Prozent dieser HIPs gebrochen werden [YSEA08a].

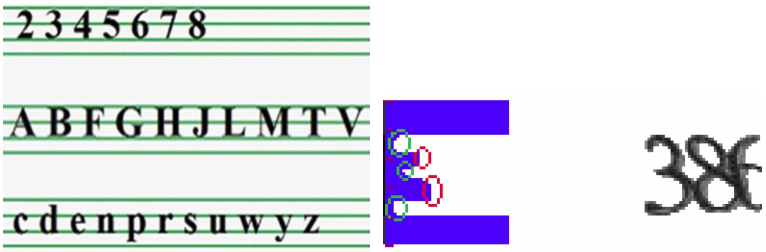
Abbildung 3.13: Durch eine gekippte Segmentierung kann auch dieses Beispiel des Yahoo HIPs aus dem Jahr 2008 segmentiert werden (aus [YSEA08a]).

Auch Huang et al. [HLBO10] wendeten ihren Angriff auf das Microsoft HIP (siehe Abschnitt 3.2.5) auf das Yahoo HIP aus dem Jahr 2008 an und konnten 45 Prozent der HIPs korrekt lösen.

Gao et al. [GWF12] versuchten bei ihrem Angriff nicht die HIPs zu segmentieren, stattdessen sollten nach und nach Zeichen erkannt und aus dem Bild gelöscht werden. Mit jedem Durchlauf müssen so immer weniger Zeichen voneinander getrennt werden. Zudem versucht dieser Angriff die Zeichen anhand ihrer Form und Position relativ zur Grundline zu identifizieren. Die Zeichen, die im Verlauf dieses Vorgangs allein stehen bleiben, werden durch einen OCR Prozess erkannt [GWF12].

Während dem Entfernen des Rauschens durch einen Schwellwertfilter kann es vorkommen, dass auch Bereiche zwischen den Zeichen gelöscht werden. An diesen Stellen entsteht somit die erste Segmentierungsstelle. Um die Grundlinie des Textes zu finden, wird diese an den untersten Zeichenpixeln ausgerichtet und entsprechend der Form des Textes gebogen. Am einfachsten können die Buchstaben p und y gefunden werden, da diese als einzige bis unter die Grundlinie ragen. Die Unterscheidung der beiden Buchstaben erfolgte anhand des geschlossenen Kreises des „p“. Abbildung 3.14(a) zeigt eine Übersicht der verwendeten Zeichen des HIPs und ihrer Ausrichtung relativ zur Grund- und den Führungslinien [GWF12].

Die jeweils ersten und letzten Zeichen wurden durch horizontale Projektion der ersten und letzten 40 Pixel und durch eine Analyse der eingeschlossenen Flächen innerhalb der Zeichen identifiziert. Manche Zeichen konnten bereits anhand ihrer Projektion erkannt werden. Dabei handelt es sich jedoch nicht um eine Projektion, wie sie bisher Verwendung fand, sondern es wird die Kontur der Zeichen abgebildet. So weist eine 3 drei Spitzen und zwei Täler auf (siehe Abbildung 3.14(b)). Andere Buchstaben wie F und B mussten zusätzlich anhand der eingeschlossenen Flächen identifiziert werden [GWF12].



(a) Die verwendeten Zeichen des HIPs und deren Position auf der Grundlinie.

(b) Projektion der Kontur des ersten Zeichens.

Abbildung 3.14: Erläuterungen zum Angriff von Gao et al. [GWF12] auf das Yahoo HIP aus dem Jahr 2012.

Wenn Zeichen miteinander verbunden werden, können ebenfalls eingeschlossene Flächen entstehen. Die Unterschiede zwischen diesen und den Flächen innerhalb der Zeichen konnten durch die Analyse der Position und Form der Flächen gefunden werden. Handelte es sich dabei um die Fläche innerhalb eines Zeichens, konnte dieses aus dem Bild entfernt und per OCR identifiziert werden. Der nächste Schritt versuchte die verbleibenden Zeichen durch vertikale Projektion zu identifizieren. Dazu erfolgte die Projektion für jeden Bereich zwischen den Führungslinien separat. Diese Projektionen wurden dann von Fenstern durchlaufen, welche die Anzahl der projizierten Pixel untersuchten. Befanden sich beispielsweise im oberen Fenster keine Pixel, jedoch im unteren Fenster weniger als acht und sind diese direkt auf der Grundlinie, so handelt es sich um ein L [GWF12].

Sollten an dieser Stelle des Angriffs noch Zeichen vorhanden sein, wird untersucht um wie viele zusammenhängende Zeichen es sich handelt. Dazu wird die Anzahl der Pixel gemessen und der Bereich der Zeichen dann in gleichem Abstand unterteilt. Dies zerschneidet zwar möglicherweise Zeichen, es konnten hierbei jedoch keine Schwierigkeiten bei dem anschließenden Erkennen per OCR festgestellt und 54 Prozent von 1000 getesteten HIPs korrekt gelöst werden [GWF12].

Viele Probleme bei der Identifikation der Zeichen stammen von der allgemein schlechten Lesbarkeit. So kann weder das Programm noch der Mensch gut zwischen c und d sowie zwischen r und n unterscheiden. Auch ist oft unklar ob es sich um eine 6 oder 8 handelt [GWF12]. Yahoo passte nach diesem Angriff das HIP in kleinem Umfang an, Gao et al. [GWF12] sind sich aber sicher, dass ihr Angriff immer noch effektiv sein wird. Zudem glauben sie, dass diese Art von Angriff auch auf die HIPs von Microsoft und Google angewendet werden kann.

Die vertikale Ausrichtung der Zeichen sollte nicht in einer ruhigen Wellenform angeordnet werden, sondern eher zufällig hin und her springen [GWF12]. Zusätzlich könnte die Verwendung von unterschiedlichen Schriftarten eine Analyse dieser Art unterbinden.

3.2.9 Andere

E-Banking

Laut Li et al. [LSK+10] verwendeten im Jahr 2010 viele Banken weltweit, sowie auch über 30 deutsche Banken HIPs um entweder Transaktionen oder ihren Login-Bereich abzusichern. Dabei wurde die Mehrzahl der HIPs im Login-Bereich eingesetzt. Ein Transaktions-HIP soll die Sicherheit der TAN-Liste wie folgt weiter erhöhen: Der Kunde gibt seine Überweisungsdaten ein und schickt diese zum Server. Dieser generiert ein HIP, das dem Browser des Kunden zugesendet wird. Dieses HIP enthält unter anderem die Daten der Überweisung. Zusätzlich wird dem Kunden an dieser Stelle mitgeteilt, welche TAN von seiner Liste er eingeben muss, um die Transaktion abzuschließen. Das von deutschen Banken verwendete HIP zeigte neben den Überweisungsdaten auch das Geburtsdatum des Bankkunden um die Echtheit des HIPs zu untermauern. Dabei waren die Textzeilen der Überweisungsinformationen zufällig angeordnet. Die Ziffern des Geburtsdatums befanden sich im Hintergrund des HIPs, wurden zufällig platziert und gedreht, und waren von unterschiedlicher Schriftart. Darüber hinaus wurde ein inhomogenes Gitter gezeichnet, welches diese Zeichen schnitt [LSK+10]. Abbildung 3.15(a) zeigt ein Beispiel dieses HIPs.

Bei dem Angriff von Li et al. [LSK+10] handelt es sich um einen Man-in-the-middle-Angriff, bei dem die vom Bankkunden eingegeben Daten auf dem Weg zum Server entsprechend geändert werden. Das HIP, das vom Server der Bank anhand der geänderten Daten erzeugt wird, muss wiederum entsprechend der ursprünglichen Daten des Bankkunden geändert werden. Der Kunde bestätigt so unwissentlich die geänderte Überweisung. Li et al. [LSK+10] befassten sich mit dem letzten Teil dieses Angriffs und versuchten die Überweisungsdaten im HIP zu ändern. Dies gelang bei allen 100 getesteten HIPs und dauerte durchschnittlich nur 250 ms. Es wurden keine trainingsbasierten Techniken verwendet, da nicht genügend Bilder der HIPs zur Verfügung standen. Um die Textzeile mit den Transaktionsdaten zu finden, konnte eine Schwachstelle ausgenutzt werden: Die beiden anderen Textzeilen enthielten entweder viele Leerzeichen oder waren fett gedruckt. Da dies in jedem Fall so war, konnte die korrekte Zeile schnell gefunden und geändert werden. Dazu musste nur der Text entfernt (siehe Abbildung 3.15(b)), der Hintergrund repariert und der neue Text eingefügt werden (siehe Abbildung 3.15(c)) [LSK+10].

Dieses HIP könnte dadurch verbessert werden, indem anstelle des Gitters zufällige Linien verwendet werden, da diese schwerer zu vervollständigen sind wenn eine ganze Zeile des Textes entfernt wurde. Zudem sollten unterschiedliche Graustufen verwendet werden, damit nicht anhand der Deckkraft des Textes unterschieden werden kann [LSK+10].

Der Angriff von Li et al. [LSK+10] auf ein Transaktions-HIP einer chinesischen Bank wird an dieser Stelle nur knapp beschrieben, da die Sicherheit aufgrund der verwendeten Farben eher gering war. Um eine Transaktion zu bestätigen, mussten lediglich die roten Ziffern in ein Textfeld eingetragen werden, denn hier kam keine TAN-Liste zum Einsatz. Da die erforderlichen Ziffern in einer anderen Farbe hervorgehoben waren, war deren Extraktion sehr einfach und sie konnten dem OCR-Prozess zugeführt werden. Abbildung 3.15(d) zeigt ein solches Beispiel. Bei einem Test konnte jedes der 100 HIPs in ca. 150 ms gelöst werden [LSK+10]. In diesem Fall bräuchte ein Angriff die Überweisungsdaten gar nicht anpassen, sondern könnte direkt die benötigten Ziffern aus dem HIP extrahieren und die Transaktion freigeben. Dem Benutzer bliebe in einem solchen Fall keine Zeit die Überweisung auf das gefälschte Konto abzuwehren.

Neben den Transaktions-HIPs testeten Li et al. [LSK+10] auch 41 verschiedene Login-HIPs und führten pro HIP-Modell 60 Versuche durch. Bis auf wenige Ausnahmen konnten alle der getesteten HIPs zu 100 Prozent geknackt werden (vgl. [LSK+10, Tabelle 1]). Abbildung 3.15(e) zeigt ein solches Login-HIP, das zu dieser Zeit von 13 deutschen Banken eingesetzt wurde und ebenfalls in allen Fällen korrekt gelöst werden konnte.

(a) Transaktions-HIP, das von deutschen Banken eingesetzt wurde.

(b) Aus dem HIP wurden bereits die Überweisungsdaten entfernt.

(c) Transaktions-HIP mit gefälschten Überweisungsdaten.

(d) Transaktions-HIP einer chinesischen Bank.

(e) HIP zur Absicherung des Login-Bereichs.

Abbildung 3.15: Verschiedene HIPs, die das E-Banking sicherer machen sollten (aus [LSK+10]).

Mobilfunkbetreiber

Ähnlich geringe Sicherheit, wie bei den Login-HIPs der Banken, entdeckte Ponec [Pon06] bei seiner Untersuchung von HIPs zweier tschechischer Mobilfunkanbieter. Diese HIPs sollten den automatisierten Versand von SMS an die Kunden der Unternehmen verhindern. Bei einem Anbieter waren dies 4,6 Millionen potentielle Spam-Empfänger [Pon06].

Das HIP von Oskar Mobil (siehe Abbildung 3.16(a)) bestand aus 5 Ziffern mit verschiedenen Schriftarten. Die Segmentierung wurde von diesem HIP in keiner Weise behindert und nach Rotation der Zeichen konnte dieses HIP durch den Vergleich mit gelernten Vorlagen zu 88 Prozent gebrochen werden. Ein großer Schwachpunk war laut Ponec [Pon06], dass bei einem erneuten Laden des HIPs dieselben Ziffern verwendet wurden, jedoch anders dargestellt. Neben der Möglichkeit das System auf diese Weise zu trainieren, konnten die HIPs so oft neu geladen werden, bis jedes Zeichen zu 100 Prozent sicher identifiziert werden konnte [Pon06].

Eurotel setzte ein HIP ein, das überhaupt keine Vorbehandlung benötigte (siehe Abbildung 3.16(b)). Es konnte ein OCR-Prozess angewandt werden, dem eine einfache Fehlerbehandlung angeschlossen war. Beispielsweise wurden erkannte Semikolon durch i ersetzt. Auf diese Art konnten 66 Prozent der HIPs gebrochen werden. Diese vergleichsweise geringe Rate ist jedoch nicht von großem Nachteil, da hier nur ein HIP pro 100 SMS gelöst werden musste [Pon06].

(a) Oskar Mobil (b) Eurotel

Abbildung 3.16: Die verwendeten HIPs zweier tschechischer Mobilfunkanbieter gegen das automatisierte Versenden von SMS (aus [Pon06]).

3.3 Angriffe auf bildbasierte HIPs

3.3.1 Asirra

Bei Asirra (siehe Abschnitt 2.3 und Abbildung 2.6) müssen aus 12 Bildern all diejenigen ausgewählt werden, die eine Katze zeigen. Dabei ist unter jedem Bild der Haustiere ein Link angebracht, unter dem weitere Informationen über das Tier sowie die Möglichkeit gegeben ist dieses zu adoptieren. Um dies in einem sinnvollen Rahmen zu halten, werden nur die Tiere angezeigt, deren Halter in einem gewissen Umkreis wohnen. Diese Tatsache könnte genutzt werden, um nur einen kleinen Teil der sehr großen Bilddatenbank für einen Angriff betrachten zu müssen [HCR09].

Golle [Gol08] verwendete für seinen Angriff auf Asirra eine *Support Vector Machine* (SVM)[3]. Zum Trainieren des Klassifizierers wurden die Beispielbilder verwendet, die von den Asirra Entwicklern [EDHS07] zum Download[4] angeboten werden. Dabei wurden zwei unterschiedliche Ansätze verfolgt: Zum einen wurden Farbinformationen pro Pixelzeile und -spalte aus den Bildern entnommen. Dabei wurde für jede Zeile oder Spalte gespeichert ob eine bestimmte Farbe darin vorkam. Zum anderen wurden die Bilder in 5x5 Pixel große Teile aufgeteilt und die Farbwerte dieser Ausschnitte klassifiziert. Hierbei wurde zudem die Textur der Ausschnitte analysiert. Für beide Ansätze wurden unterschiedlich große Bildsätze verwendet, die Zahlen der Resultate beziehen sich jedoch immer auf eine Größe des Trainingssets von 8000 Bildern, da hier die Genauigkeit am Größten war. Dem gegenüber standen 2000 Bilder, die klassifiziert wurden. Die SVM konnte mit Hilfe der Farbwerte pro Zeile und Spalte 77 Prozent der Testbilder korrekt zuordnen. Der Ansatz, bei dem Texturdaten klassifiziert wurden, erreichte 80 Prozent, eine Kombination der beiden Methoden konnte 82,7 Prozent der Bilder korrekt zuordnen. Da pro HIP 12 Bilder verwendet wurden, ergibt sich eine Gesamteffektivität von 10 Prozent $(0,827^{12} = 0,102)$ [Gol08].

Golle [Gol08] berichtete von Bildern, die Katzen und Hunde oder gar kein Tier zeigten. Diese wurden für die Tests aussortiert, können aber den Erfolg eines echten Angriffs und auch die Gebrauchstauglichkeit senken. Die Entwickler des HIPs [EDHS07] diskutierten Vorschläge, wie Asirra verbessert werden könnte: Löst ein Benutzer eine Aufgabe fast richtig (11 von 12 korrekt), so darf er es noch einmal versuchen. Löst der Benutzer nun wieder nur 11 von 12 richtig, erhält er trotzdem Zugang. Dies sollte die Benutzerfreundlichkeit erhöhen [EDHS07]. Durch die Umsetzung dieser Erweiterung würde die Erfolgswahrscheinlichkeit des Angriffes von Golle auf 38% steigen [Gol08]. Eine andere Erweiterung sah vor, Benutzer, die dieses HIP oft falsch lösen, mehrere HIPs hintereinander lösen zu lassen [EDHS07]. Golle [Gol08] bescheinigte seinem Angriff bei zwei aufeinander folgenden HIPs nur noch 1 Prozent Erfolg.

[3] Eine *Support Vector Machine* unterteilt eine Menge von Objekten so in Klassen, dass um die Klassengrenzen herum ein möglichst breiter Bereich frei von Objekten bleibt (http://de.wikipedia.org/wiki/Support_Vector_Machine).

[4] http://research.microsoft.com/en-us/projects/asirra/corpus.aspx

Der Angriff von Golle ist resistent gegen eine mögliche Verzerrung der Bilder, da in diesem Fall der Farbklassifizierer immer noch arbeitet. Ebenso ist er resistent gegen die Verwendung von Graustufenbildern, da hier der Texturklassifizierer immer noch korrekte Ergebnisse liefert [Gol08].

Hernandez-Castro et al. [HCRS09] schlossen allein durch die Bildgröße der Katzen- bzw. Hundebilder zu 57,8 Prozent auf das korrekte Tier. Diese Sicherheitslücke kann durch einen entsprechenden Filter vermieden werden [HCRS09]. Eine andere Möglichkeit wäre, das HIP serverseitig zusammen zu setzen und dann als eine Datei zu übertragen. Mit 57,8 Prozent Erfolg eignet sich dieser Ansatz jedoch nicht für einen Einsatz. Denn die Erfolgschancen ein HIP mit 12 Bildern korrekt zu lösen betragen in diesem Fall nur 0,1 Prozent ($0,578^{12} = 0,001$).

3.3.2 Avatar Captcha System

Das Avatar Captcha System zeigt in zwei Reihen je sechs Graustufenbilder von Gesichtern. Dabei handelt es sich zum einen um Fotografien von realen Gesichtern aus einer Datenbank und zum anderen um generierte Gesichter, wie in Abbildung 3.17 zu sehen ist. Der Benutzer muss die 12 Bilder mittels Checkboxen als echte bzw. generierte Gesichter klassifizieren. Nur wenn alle 12 Bilder korrekt bewertet wurden, erlangt der Benutzer Zugang zum geschützten System. Die Entwickler des Avatar Captcha System [DPY12] fanden bei ihrer Evaluation heraus, dass Menschen diese Aufgabe in 63% der Fälle lösen können. Die Erfolgschancen eines angreifenden Programms schätzten die Entwickler auf 0,02%. Dabei gingen sie von einem System aus, das für jedes Bild rät ob es generiert wurde oder nicht [DPY12].

Abbildung 3.17: Beispiel eines HIPs des Avatar Captcha System (aus [DPY12]).

Für ihren Angriff verwendeten Korayem et al. [KMCY12] ein Demoset, das aus jeweils 50 Bildern realer und generierter Gesichter bestand. Dieses Set wurde von den Entwicklern zur Verfügung gestellt. Korayem et al. [KMCY12] erprobten verschiedene Techniken und erreichten bis zu 100 Prozent Erfolg. Dies gelang durch *feature extraction* aus allen Pixeln des jeweiligen Bildes. Da die Bilder eine Abmessung von 50 x 75 Pixel aufwiesen konnte so ein Vektor mit 3750 Dimensionen erstellt werden. Eine Klassifizierung mit einem LibLinear Klassifizierer erzielte die obengenannten 100% und ist somit erfolgreicher als die Testpersonen. Ein weiterer Angriff nach dem gleichen Verfahren wurde von Yamasaki und Chen [YC12a] durchgeführt. Während Korayem et al. [KMCY12] nicht beschreiben, wie viele Samples für das Training verwendet wurden, geben Yamasaki und Chen [YC12a] für ein Trainingsset von 10 Bildern pro Klasse eine Erfolgsquote von 95% und bei 40 Bildern 99,5% an.

Zudem verwendeten Yamasaki und Chen [YC12a] ein Verfahren, bei dem das jeweilige Bild durch das Bilden von Clustern in mehrere Layer unterteilt wird, für die jeweils ein Merkmalsvektor gebildet wird. Hier wurde eine besondere Form dieses Verfahrens angewendet, bei dem in die Berechnung der

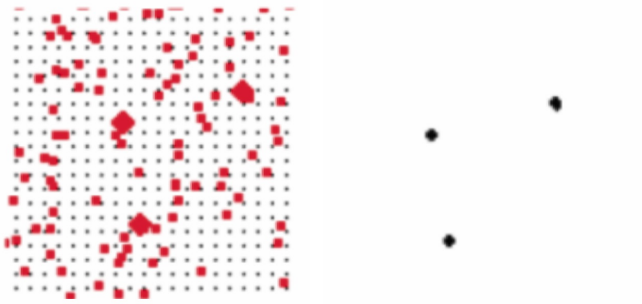
Merkmalsvektoren die Durchschnittswerte der anderen Layer einfließen [YC12b]. Dieses Verfahren wird *sparse coding + spatial pyramid matching + spatial statistics* genannt und erzielte mit 40 Trainingsbildern ebenfalls eine Erfolgsquote von 100% [YC12a].

Korayem et al. [KMCY12] empfehlen die Kombination aus Rotation, hinzufügen von Rauschen und das Verdecken von besonders eindeutigen Bildbereichen. Ein Test zeigte jedoch, dass die Kombination aus allen 3 Techniken den erfolgreichsten Klassifizierer nur auf 14% Erfolg pro HIP senken konnte. Die Verfasser geben weiter zu bedenken, dass die Gebrauchstauglichkeit der so bearbeiteten Bilder nicht getestet wurde und mit Sicherheit unterhalb der 63% Erfolg der unbearbeiteten Bilder liegt [KMCY12].

3.3.3 Drawing CAPTCHA

Shirali-Shahreza und Shirali-Shahreza [SSSS06] entwickelten ein HIP für Geräte mit Touch-Display. Der Hintergrund des HIPs bestand aus grauen Punkten von der Größe genau eines Pixels. Eine erhebliche Anzahl von größeren Quadraten und wenigen noch größeren Rauten stellten den Vordergrund dar, wie in Abbildung 3.18(a) zu sehen ist. Der Benutzer musste alle Rauten miteinander verbinden, dabei spielte die Reihenfolge keine Rolle. Die Sicherheit des HIPs beruhte auf der Tatsache, dass Menschen besser darin sind, unter Rauschen bestimmte Formen zu finden. Das Zeichnen von Linien ist auf einem Touch-Display sehr einfach, und besser geeignet als auf einer Bildschirmtastatur Zeichen einzugeben [SSSS06].

Leider sind die Punkte oft zu klein um diese genau treffen zu können, zudem bieten die gewählten Formen Angriffspunkte für bildverarbeitende Angriffe [LHBL11]. Für ihren Angriff wandelten Lin et al. [LHBL11] das Bild zuerst in Schwarz-Weiß Farben um. Danach wurden Weichzeichnungsfilter eingesetzt, die das Rauschen und einen Großteil der Quadrate entfernten. Nach diesem Schritt waren nur noch die Rauten und eventuell vorhandene Gruppen von Quadraten zu sehen (siehe Abbildung 3.18(b)). Die größten der verbleibenden Objekte wurden als Rauten behandelt und verbunden. Bei einem Test mit 20 generierten HIPs konnten 75% korrekt gelöst werden [LHBL11].



(a) Ein Beispiel des Drawing CAPTCHA (aus [LHBL11]).

(b) Das linke HIP nach der Bearbeitung durch Lin et al. [LHBL11].

Abbildung 3.18: Darstellung des Angriffs von Lin et al. [LHBL11] auf das Drawing CAPTCHA HIP.

3.4 Angriffe auf audiobasierte HIPs

Kochanski et al. [KLS02] zeigten, dass Menschen im Bereich der Erkennung von Sprache, die mit Störgeräuschen belastet ist, besser sind als Maschinen. Auf dieser Erkenntnis basieren alle bis heute entwickelten Audio-HIPs. Diese sind mittlerweile als Alternative zu den textbasierten HIPs weit verbreitet und sollen die Barrierefreiheit für Menschen mit Sehbehinderung garantieren. Fast alle dieser HIPs bestehen aus einer Sequenz von gesprochenen Buchstaben oder Ziffern, die mit Rauschen oder anderen Störgeräuschen verschiedenster Art verzerrt werden [BBP+11]. Da die folgenden Angriffe immer auf mehrere HIPs erfolgreich angewendet wurden, erfolgt in diesem Abschnitt keine Unterteilung auf HIP-Ebene.

Tam et al. [TSHA08] führten ihren Angriff in zwei Phasen durch. In der ersten Phase wurden die Ausschnitte, in denen ein Zeichen zu hören war, extrahiert und in der zweiten Phase versucht, dieses Zeichen durch maschinelles Lernen zu identifizieren. So konnten die Audio-HIPs von Google und reCAPTCHA zu 67% bzw. 45% gebrochen werden. ReCAPTCHA führte daraufhin eine Anpassung durch und verwendet seitdem *semantic noise* (siehe Abschnitt 2.2) [TSHA08].

Bursztein et al. [BBP+11] führten ebenfalls einen Angriff in zwei Phasen durch und waren in der Lage folgende Audio-HIPs zu brechen: Microsoft (49%), Yahoo (45%) und eBay (83%). ReCAPTCHA konnte aufgrund der Verbesserungen nur zu 1% gebrochen werden. Diese Zahl scheint für das automatisierte Lösen sehr gering, ist aber ähnlich gut wie die Erfolgsquote von Menschen (siehe dazu Abschnitt 4.3.1) [BBF+10]. Für diesen Angriff benötigte das System ein Trainingsset aus 300 Audiosamples und 20 Minuten Trainingszeit. Nach dem Training konnten 10 Audio-HIPs pro Minute auf einem einzigen PC gelöst werden. Diese Ergebnisse zeigen, dass ein nicht kontinuierliches Audiosignal für den Einsatz in HIPs als unsicher zu betrachten ist. Bursztein et al. [BBP+11] gehen davon aus, dass es nicht möglich ist mit aktuellen Methoden ein sicheres Audio-HIP zu erstellen, welches zugleich von Menschen gelöst werden kann. Sicher bedeutet in diesem Fall eine maximale Erfolgsquote von 0,01% für automatisierte Angriffe.

Um diese HIPs zu verbessern wird deshalb die Verwendung von *semantic noise* empfohlen [BBP+11]. Die Untersuchten Audio-HIPs verwendeten alle bis auf reCAPTCHA ein kontinuierliches Rauschen. Bursztein et al. [BBP+11] sind der Meinung, dass Störgeräusche, welche ebenfalls aus Sprache bestehen, das maschinelle Lernen sehr erschweren, den Menschen aber kaum negativ beeinflussen.

Bei der Untersuchung von Bursztein et al. [BBP+11] konnten die Audio-HIPs von Google nicht getestet werden, da die Verfasser aufgrund der schweren Verständlichkeit der Zeichen nicht mit Sicherheit sagen konnten, ob die von ihrem Programm identifizierten Zeichen korrekt sind und gehen deshalb davon aus, dass dieses HIP für den Einsatz am Benutzer zu ineffektiv ist.

Es wird vermutet, dass Audio-HIPs deshalb so leicht zu brechen sind, da im menschlichen Gehirn der Bereich für die Verarbeitung von Audio weniger Platz einnimmt als der von visuellen Eindrücken [BBP+11]. Auch ist der Stand der Technik im Bereich der Audioverarbeitung und das maschinelle Lernen sehr fortgeschritten. Aus diesem Grund ist der Unterschied zwischen Mensch und Maschine in diesem Bereich eher gering [BBF+10, BBP+11].

3.5 Angriffe auf spielbasierte HIPs

Leider konnten hierzu keine wissenschaftlichen Publikationen gefunden werden. Dies könnte daran liegen, dass diese Form von HIPs noch zu neu oder zu schwer zu brechen ist. Es wurde jedoch von einem vermeintlich erfolgreichen Angriff auf das bereits erwähnte HIP von Are You a Human berichtet (siehe Abschnitt 2.4) [MSG+13]. Dieser Angriff wird hier nur kurz erläutert, da dessen Effektivität nicht eindeutig geklärt werden konnte.

Für den Angriff wurde eine Anwendung entwickelt die durch Ausprobieren verschiedene Gegenstände nimmt und auf die Ablageposition verschiebt [SPA12]. Da hierbei auch falsche Gegenstände bewegt werden, wird der Benutzer korrekt als Bot erkannt. Eine Weiterentwicklung der Anwendung kann nun gezielt die richtigen Gegenstände auswählen und platzieren. Dies gelingt jedoch bisher nur bei einem bestimmten Spiel. Die Seite muss so lange neu geladen werden, bis das richtige Spiel erscheint [SPA12]. Da nicht bekannt ist, wie viele unterschiedliche Spiele in dem Are You a Human HIP vorhanden sind, kann an dieser Stelle leider keine Aussage über die Erfolgschancen dieses Angriffes getroffen werden. Zudem gaben Entwickler von Are You a Human zu bedenken, dass die Auswertung des Benutzerverhaltens serverseitig vorgenommen wird und immer weiter verbessert werden kann [Are14c]. Zusätzlich könnte erfasst werden, ob bestimmte Spiele häufiger gelöst werden als andere. Mit diesem Wissen sollte dieser Angriff mit simplen Mitteln erfolgreich abgewehrt werden können.

3.6 Abgeleitete Richtlinien für das Erstellen von sicheren HIPs

Die eigentliche Aufgabe von HIPs ist es, den Anbieter eines Dienstes oder den Betreiber der Internetseite vor automatisierten Angriffen zu schützen [Bri11]. Bei den aufgeführten HIPs des E-Banking und der Mobilfunkbetreiber sollen jedoch hauptsächlich die Kunden vor Angreifern bzw. Spam geschützt werden. Aus diesem Grund erscheint es unverständlich, dass gerade hier derart unsichere HIPs eingesetzt wurden. Die Tatsache, dass heute zumindest von den meisten deutschen Banken keine HIPs mehr eingesetzt werden scheint die logische Konsequenz daraus zu sein. Denn bietet ein HIP nicht ausreichend Schutz, unabhängig davon was geschützt werden soll, so ist kein HIP einzusetzen die bessere Wahl, da auf diese Weise die Benutzer nicht unnötig belästigt werden [Bri11, GWLY11].

Soll jedoch ein HIP zum Einsatz kommen, so muss es gewisse Anforderungen unbedingt erfüllen. Für menschliche Benutzer sollte es möglich sein, das HIP in mehr als 80 Prozent der Versuche erfolgreich zu lösen, für eine Maschine muss diese Quote jedoch weniger als 0,01 Prozent betragen [CS05]. Um dies zu erreichen, können die folgenden Richtlinien verwendet werden, die aus der Analyse der zuvor beschriebenen Angriffe abgeleitet wurden.

Bei einem textbasierten HIP muss die Auswahl der Zeichen die oberste Priorität haben. Zum einen sollte nie eine feste Anzahl von Ziffern verwendet werden, ebenso wenig sollte eine Schriftart eingesetzt werden, bei der alle Zeichen dieselbe Breite aufweisen [YSEA08a]. Denn diese Faktoren vereinfachen den Segmentierungsvorgang erheblich (vgl. Angriff auf das Microsoft HIP, Abschnitt 3.2.5). Zum anderen ist die Auswahl der Zeichen maßgebend. Einerseits welche Art von Zeichen überhaupt Verwendung finden soll, wie beispielsweise Zahlen und Buchstaben in Groß- und Kleinschreibung. Hier gilt, je mehr Zeichen verwendet werden, desto sicherer kann das HIP sein [HGH08]. Jedoch sinkt damit auch die Gebrauchstauglichkeit aufgrund von ähnlichen und somit leicht zu verwechselnden Zeichen. Andererseits muss entschieden werden ob Zufallskombinationen dieser Zeichen verwendet werden sollen oder bekannte Wörter. Auch hier gilt zwischen Sicherheit und Lesbarkeit abzuwägen,

denn ist die Auswahl an Wörtern stark begrenzt, so stellt dies eine erhebliche Schwachstelle dar, wie der Angriff auf Gimpy (siehe Abschnitt 3.2.3) und Captchaservice.org (siehe Abschnitt 3.2.2) zeigte.

Unabhängig von den verwendeten Zeichen sollten diese zufällig platziert werden, sodass die Position der aufeinander folgenden Zeichen nicht abschätzbar ist, wie dies bei dem Yahoo HIP der Fall war (siehe Abschnitt 3.2.8). Ebenso sollte, sofern Verzerrung eingesetzt wird, nicht das gesamte HIP mit globaler Verzerrung behandelt werden, da diese Verzerrung leicht rückgängig gemacht werden kann (vgl. Angriff auf Gimpy, Abschnitt 3.2.3). Stattdessen kann jedes Zeichen für sich global verzerrt werden, um nicht anhand eines Zeichens die Verzerrung des gesamten HIPs zurückrechnen zu können. Die verschiedenen Möglichkeiten, eine Verzerrung anzuwenden wurden bereits in Abschnitt 3.2 beschrieben und in Abbildung 3.1 dargestellt.

Um das Identifizieren der Zeichen zu verhindern, muss dafür gesorgt werden, dass das Bild nicht in die einzelnen Zeichen zerlegt werden kann. Die Ansätze die hierfür Rauschen in Form von Linien und Bögen verwendeten, wurden bereits alle erfolgreich gebrochen (siehe Abschitt 3.2.1, 3.2.5, 3.2.7, 3.2.8 und 3.2.9). Auch der Versuch eine Schriftart zu verwenden, bei der die Zeichen aus mehreren Teilen bestehen, führte nicht zum Erfolg, wie von Kluever [Klu08a] mit dem Angriff auf das PayPal HIP zeigen konnte (siehe Abschnitt 3.2.7). Aus diesem Grund scheint nur die Überlappung der Zeichen einen ausreichenden Schutz zu bieten, wie dies von Google seit dem Jahr 2008 erfolgreich demonstriert wird [GWF12]. Hierbei muss jedoch beachtet werden, dass darunter ebenfalls die Lesbarkeit leiden kann, da sich so Buchstabenpaare und einzelne andere Buchstaben sehr ähneln können. Wird Rauschen eingesetzt, so muss dieses ebenfalls rein zufällig platziert werden, da sonst anhand der relativen Position darauf geschlossen werden kann ob es sich um Rauschen oder ein Zeichen handelt, wie dies bei dem Angriff auf das Microsoft HIP geschehen ist (siehe Abschnitt 3.2.5). Zudem muss das Rauschen dieselbe Farbe wie die Zeichen aufweisen, da es sonst sehr einfach herausgefiltert werden kann. Siehe hierzu: Angriff auf das PayPal HIP, Abschnitt 3.2.7.

Soll ein Hintergrund verwendet werden, so sollte dieser ebenfalls zufällig generiert werden. Die Angriffe zeigen jedoch, dass Hintergründe eine sehr schlechte Verteidigungsstrategie sind und eher die Lesbarkeit für den Menschen beeinträchtigen. Beispiele hierfür sind Gimpy (Abschnitt 3.2.3), PayPal (Abschnitt 3.2.7) und das E-Banking HIP (Abschnitt 3.2.9). Ein Hintergrund kann nur positiv auf die Sicherheit wirken, wenn er auch dieselbe oder eine sehr ähnliche Farbe der Zeichen enthält [HGH08]. Sonst kann der Hintergrund ganz einfach herausgefiltert werden. Es ist denkbar den Hinter- und Vordergrund in den gleichen Farben zu gestalten, die gleich gefärbten Zeichen jedoch an einer anderen Stelle zu platzieren. Es ist auch denkbar die Zeichen in verschiedenen Farben zu gestalten, dass diese nicht durch *color filling* erkannt werden können [HGH08]. Wie sich dies auf die Lesbarkeit auswirkt sollte im Einzelfall genau untersucht werden. Wichtig ist jedoch auch hier, dass die Farbmuster keinem vordefinierten Schema folgen.

Einige Angriffe weisen eine hohe Zahl an Ausnahmen auf, die behandelt werden müssen. Beispielsweise musste von Huang et al. [HLBO10] darauf geachtet werden ob es sich um die Zeichen 0, O, D, B oder 8 handelt (siehe Abschnitt 3.2.5). Hierin könnte Potential stecken, denn dadurch würde ein Angriff möglicherweise zu komplex um ihn an kleine Änderungen des HIPs anzupassen.

Wie bereits erwähnt sollte der Aufbau des HIPs rein zufällig geschehen. Damit sind einerseits die Zeichen sowie das Rauschen und dessen Position, andererseits jedoch auch die Komposition des HIPs gemeint. Am Beispiel des Megaupload HIPs wurde dies besonders anschaulich (siehe Abschnitt 3.2.6). Hier wurden die Zeichen immer in der gleichen Reihenfolge und im selben Winkel und Abstand zusammengesetzt. Eine Variation dieser Eigenschaften hätte den Angriff sicher erschwert [SEAYM10].

Auf der anderen Seite ist es wichtig die Varianz aus den verwendeten Aufgaben zu entfernen [HCR09]. Dies ist jedoch nur ein Problem wenn ein Pool von Aufgaben verwendet wird und nicht wenn diese zufällig generiert werden[5]. Ein Beispiel hierfür ist Asirra (siehe Abschnitt 2.3), bei dem von Hernandez-Castro et al. [HCRS09] ein Unterschied der Dateigröße zwischen den Katzen- und Hundebildern entdeckt werden konnte. Ebenfalls sollte ein HIP nicht in Abhängigkeit des Benutzers generiert werden. Hierfür ist Asirra wieder ein gutes Beispiel, denn hier wurden nur Tiere angezeigt, die sich in der Nähe des Benutzers befinden. Nutzt ein Hacker dieses Wissen, müsste er nur einen kleinen Teil der Bilddatenbank betrachten [HCR09].

Zugleich sollte auf eine ausgewogene Verteilung der Antworten geachtet werden. So konnte das HIP MathCAPTCHA gebrochen werden, bei welchem eine Rechenaufgabe gelöst werden musste. Da diese Aufgaben von Menschen im Kopf gelöst werden sollten, war das Ergebnis meist eine kleine, ganze Zahl. Mit diesem Wissen konnten bis zu 93% der Aufgaben durch raten gelöst werden [HCR09, HCR10].

In Audio-HIPs sollte *semantic noise* verwendet werden, da hiermit scheinbar Angriffe abgewehrt werden können [BBP+11]. Andernfalls müsste möglicherweise das Audiosample weiter bearbeitet werden, was das Lösen für den Menschen weiter erschwert.

Zudem sollte es einem Angreifer schwierig gemacht werden, herauszufinden ob er die Aufgabe korrekt gelöst hat [HCR09]. Dies wurde im HIP von Are You a Human (siehe Abschnitt 2.4) versucht. Zwar beendet sich das Spiel wenn alle Gegenstände korrekt platziert wurden, jedoch ist an dieser Stelle noch nicht klar ob das Spiel erfolgreich abgeschlossen wurde. Dies wird dem Benutzer erst nach einem weiteren Klick, der die Auswertung auf dem Server anstößt, angezeigt. Wie auch bei den Alternativen zu HIPs (siehe Abschnitt 2.6) bleibt dadurch dem Angreifer teilweise verborgen, ob sein Angriff wirksam ist.

Die wiederverwendbaren Angriffe zeigen, dass sich manche HIPs in ihrer Ausführung nicht wesentlich unterscheiden. So konnten viele Angriffe, die eigentlich für das Microsoft HIP entwickelt wurden, ebenso auf das Yahoo HIP angewendet werden. Das könnte an den begrenzten Mitteln liegen, die für das Verhindern der Segmentierung verfügbar und nutzbar sind, da in beiden HIPs Rauschen in Form von Linien eingesetzt wurde. Vor allem im Bereich der textbasierten HIPs gilt es deshalb neue Techniken gegen automatisierte Angriffe zu finden, da im schlimmsten Fall ein Angriff auf alle HIPs angewendet werden könnte.

Computer sind bereits heute bei der Erkennung von deformierten Zeichen dem Menschen voraus [CLSC05b]. Da die Fähigkeiten hier in Zukunft sicher zunehmen werden, ist fraglich wie lange textbasierte HIPs noch sinnvoll eingesetzt werden können. Für sichere audiobasierte HIPs ist es erforderlich die Audiosamples derart zu entstellen, dass es für Menschen fast unmöglich ist, diese zu verstehen. Es muss die Zukunft zeigen, wie der Einsatz von *semantic noise* aus dieser Spirale immer stärker verzerrten Audiosamples hinausführen kann.

Da auf spielbasierte HIPs keine erfolgreichen Angriffe bekannt sind, könnte hier ein falscher Eindruck von Sicherheit entstehen. Tatsächlich muss diese Sicherheit erst noch bewiesen werden. Aufgrund der größeren Komplexität der Aufgaben kann jedoch davon ausgegangen werden, dass Angriffe hierauf aufwändiger sein müssen und somit nicht in jedem Fall rentabel sein werden. Unter anderem aus diesem Grund wurde diese Form für das HIP, das im Rahmen dieser Arbeit entwickelt und evaluiert wurde, gewählt.

[5] Die Tatsache, dass ein Computer allein keinen perfekten Zufallsgenerator bietet, ist für diesen Verwendungszweck nicht ausschlaggebend.

4 Gebrauchstauglichkeit

Als Pendant zu den Aspekten aus dem vorherigen Kapitel, die es in Bezug auf die Sicherheit zu beachten gilt, werden in diesem Kapitel Aspekte in Bezug auf die Gebrauchstauglichkeit und Benutzerfreundlichkeit von HIPs beschrieben. Dabei werden sowohl technologiespezifische als auch allgemeingültige Aspekte thematisiert. Da bild- und spielbasierte HIPs sehr unterschiedlich und spezifisch sind, können an dieser Stelle keine gesonderten Richtlinien vorgegeben werden. Entstehen bei solchen HIPs keine sprachlichen und kulturellen Barrieren, stellen sie eine sehr gute Alternative gegenüber den Text- und Audiobasierten dar. Dazu ist die Beachtung aller sonst üblichen Gesichtspunkte der Benutzerfreundlichkeit jedoch unerlässlich.

Für die Ableitung der bestimmenden Aspekte werden in den folgenden Abschnitten bekannte HIPs untersucht und ihre Eigenschaften im Hinblick auf die Benutzerfreundlichkeit bewertet. Zuvor werden jedoch Eigenschaften beschrieben, die bei der Entwicklung aller HIPs beachtet werden sollten. Am Ende des Kapitels werden Aspekte der Gebrauchstauglichkeit von HIPs im mobilen Einsatz beleuchtet und alle gewonnenen Erkenntnisse im letzten Abschnitt zusammengefasst.

4.1 Allgemein

Neben dem Design des Interfaces eines HIP ist auch der Arbeitsaufwand für den Benutzer maßgeblich. Dies schließt sowohl die Schwierigkeit der Aufgabe, als auch die Zeit, die der Benutzer für das Lösen der Aufgabe aufbringen muss, ein. Beispielsweise wurden bei dem HIP MathCAPTCHA nur Rechenaufgaben gestellt, die relativ einfach zu bewältigen waren, jedoch war die Lösung meist eine ganze Zahl zwischen null und zehn [HCR10]. Durch die Wahl dieser Aufgaben wurde einerseits die Gebrauchstauglichkeit sichergestellt, jedoch gleichzeitig auch eine Sicherheitslücke verursacht. Dies zeigt wie schwer es ist, ein sicheres und gleichzeitig gebrauchstaugliches HIP zu entwickeln.

Ein Benutzer sollte die gestellte Aufgabe in fünf bis zehn Sekunden korrekt lösen können [BBF+10]. Wie wichtig es ist, dass die Aufgaben in jedem Fall lösbar sind zeigt die Tatsache, dass ein Benutzer durchschnittlich eine Sekunde länger benötigt, wenn er die Aufgabe falsch löste [BBF+10]. Die gefühlte Zeit dürfte in einem solchen Fall noch länger sein. Deshalb ist es nötig, lösbare Aufgaben zu stellen um die Benutzer nicht weiter zu frustrieren. Dieses Zeitlimit wird im Bereich der audiobasierten HIPs fast nie eingehalten, denn hier weisen die meisten der verwendeten Audiosamples eine Laufzeit von mehr als zehn Sekunden auf. Spitzenreiter war hier im Jahr 2010 das Google HIP mit einer Laufzeit von durchschnittlich etwa 37 Sekunden [BBF+10].

Aufgrund der hohen Zahl von täglich gelösten HIPs summieren sich diese kleinen Werte jedoch zu sehr großen Zahlen auf: Während im Jahr 2004 weltweit etwa 150 000 Stunden pro Tag für das Lösen von HIPs aufgewendet wurden, so waren dies 2012 bereits über 550 000 Stunden pro Tag, also fast 63 Jahre [reC04, Sut12]. Das HIP reCAPTCHA (siehe Abschnitt 2.1) wurde entwickelt um zumindest einen Teil dieser Zeit für etwas Sinnvolles zu verwenden. Deshalb werden seit der Entwicklung des HIPs im Jahr 2007 Wörter für den Texterkennungsprozess von Google Books dadurch identifiziert [HCR09, reC04].

Ein weiteres grundlegendes Problem ist die Größe der HIPs. Zwar kann an dieser Stelle keine genaue Aussage über die notwendige Größe gemacht werden, es sollte jedoch möglich sein, das HIP auf jedem internetfähigen Gerät betrachten zu können. Leider ist dies mit den heute verwendeten HIPs nicht immer möglich, da diese zu groß für das Display von Smartphones sind [RC13, MSG+13]. Jedoch sollte auch eine Mindestgröße eingehalten werden, da die HIPs sonst möglicherweise zu klein angezeigt werden und der Benutzer zuerst zoomen muss.

4.2 Textbasierte HIPs

Textbasierte HIPs sind am weitesten verbreitet und bieten viele Vorteile: Sie sind intuitiv für alle Menschen weltweit und grenzen wenige aus, da viele Menschen lateinische Buchstaben lesen können [YSEA08a]. Es ist jedoch zu bedenken, dass es Benutzer geben könnte, denen nicht klar ist was zu tun ist. In einem solchen Fall ist es tiefgreifend, in welcher Sprache die Erläuterungen verfasst sind. Auch wurden bei dieser Aussage Blinde und Analphabeten außen vor gelassen. Jedoch sind textbasierte HIPs durch ihre Verbreitung mittlerweile fast jedem Internetnutzer bekannt und benötigen deshalb eigentlich keine Anleitung mehr. Die Gestaltung von textbasierten HIPs kann zwischen den Dimensionen Verzerrung, Inhalt und Darstellung unterschieden werden [YSEA08b]. Aus diesem Grund wurde dieser Abschnitt ebenso strukturiert.

4.2.1 Verzerrung

Bei Verzerrung ist die Methode und die Intensität ausschlaggebend [YSEA08b]. Die lokale und globale Anwendung dieser Technik wurde bereits in Abschnitt 3.2 beschrieben. Das Verschieben der Zeichen nach oben oder unten, sowie die Rotation oder Skalierung, als auch das Deformieren von Zeichen stört die Rezeption des Menschen nicht [CLSC05a]. Allerdings kann damit allein kein HIP sicher gestaltet werden. Denn je mehr ein Zeichen deformiert wurde, umso besser kann ein Computerprogramm dieses im Vergleich zum Menschen erkennen [CLSC05b]. Deshalb ist es wichtig, dass diese Technik beispielsweise mit Überlappung so kombiniert wird, dass es einem angreifenden Computerprogramm nicht gelingt ein Zeichen zu extrahieren. Verzerrung allein kann also ein HIP nicht sicherer machen. Es wird dadurch einzig für die Benutzer schlechter lesbar [CLSC05b].

Denn durch Überlappung und Deformation der Zeichen kann es dazu kommen, dass diese nicht mehr eindeutig zu identifizieren sind. Beispiele hierfür sind: O, 0; 5, s, S; 6, G, b; 2, Z, z; 1, I, l [YSEA08b]. Aus diesem Grund werden bei vielen textbasierten HIPs bestimmte Zeichen nicht verwendet. Das Problem hierbei ist, dass der Benutzer nicht weiß welche Zeichen überhaupt verwendet werden. Es darf demnach kein Zeichen eines solchen Paares verwendet werden, wodurch die Anzahl der möglichen Kombinationen stark eingeschränkt werden kann.

Neben einzelnen Zeichen stellen auch Zeichengruppen ein Problem dar. Durch das mittlerweile sehr enge Setzen der Zeichen, um eine Segmentierung zu verhindern, werden Verwechslungen immer wahrscheinlicher. Beispiele hierfür sind: vv, w; cl, d; nn, m [YSEA08b]. Sechs Prozent der HIPs, die von Google im Jahr 2008 eingesetzt wurden, enthielten solche problematischen Zeichengruppen, von denen einige in Abbildung 4.1 gezeigt sind [YSEA08b]. Diese Schwierigkeiten können noch verstärkt werden, wenn Benutzer andere Zeichen als die lateinischen gewohnt sind. Beispiele hierfür sind Benutzer, die sonst chinesische oder kyrillische Zeichen verwenden. Diese könnten durch zu stark deformierte Zeichen fehlgeleitet werden [YSEA08b].

Abbildung 4.1: Nicht eindeutige Zeichen, welche durch Verzerrung und Überlappung entstehen können (aus [YSEA08b]).

4.2.2 Inhalt

Bei textbasierten HIPs beschreibt diese Gestaltungsdimension die Menge von Zeichen, die zum Einsatz kommen können [YSEA08b]. Es geht also um die Auswahl der Zeichen, die von den Entwicklern eines HIPs getroffen wurde. Diese könnte beispielsweise dadurch eingeschränkt worden sein, dass sich bei der verwendeten Schriftart Zeichen zu ähnlich sehen, wie bereits im vorherigen Abschnitt beschrieben.

Werden für ein HIP nur Zahlen verwendet, kann ein Benutzer 98 Prozent dieser HIPs korrekt lösen [YSEA08b]. Diese Zahl sinkt auf 82 Prozent wenn zusätzlich Klein- und Großbuchstaben von a bis z verwendet werden [Sut12]. Jedoch steigt damit auch die Sicherheit des HIPs, da es eine größere Menge von Zeichen gibt, die es zu identifizieren gilt. Grundlegend sollten deshalb so viele Zeichen wie möglich verwendet werden [GWLY11].

Auch die Anzahl der jeweils sichtbaren Zeichen ist extrem wichtig, denn wie gezeigt wurde, kann durch eine feste Anzahl von Zeichen ein HIP unter Umständen leicht gebrochen werden. Allerdings sinkt mit steigender Anzahl der Zeichen auch die Chance alles korrekt zu identifizieren. Diese Tatsache vereitelt möglicherweise einen Angriff, kann aber für Benutzer eine weitere Hürde darstellen [YSEA08b]. Deshalb sollten Zeichenketten zufälliger Länge verwendet werden, die jedoch nicht zu lang oder zu kurz sein sollten. Als Richtwert können fünf bis acht Zeichen angenommen werden.

Ebenso müssen sich Entwickler eines HIPs darauf einigen, ob zufällige Zeichen oder Wörter verwendet werden sollen [YSEA08b]. Auch hier muss genau abgewägt werden, denn während Wörter das Lösen für einen Benutzer wesentlich vereinfachen können, so gilt dies möglicherweise auch für einen Angriff [YSEA07]. Deshalb ist unbedingt darauf zu achten, eine sehr große Anzahl von unterschiedlichen Wörtern zu verwenden. Der Benutzer wird sich die verschiedenen Wörter ohnehin nicht merken, und für den Angreifer wird es so schwieriger das korrekte Wort zu finden. Auch in diesem Fall sollten Wörter mit unterschiedlicher Länge verwendet werden, das vorgeschlagene Limit von acht Zeichen ist hierbei jedoch fakultativ.

Werden Wörter verwendet, muss der Benutzer nicht zwingend jeden Buchstaben eindeutig erkannt haben um das HIP korrekt lösen zu können [YSEA08b]. Dies gilt jedoch nur wenn der Benutzer das gesuchte Wort kennt, er also der verwendeten Sprache mächtig ist [BBF+10]. Da aus Sicherheitsgründen zufällige Zeichenfolgen vorzuziehen sind, setzt Google beim Erzeugen der HIPs einen Generator ein, der Zeichenfolgen erzeugt, die zwar kein Wort darstellen, jedoch phonetisch ähnlich klingen [BBF+10, GWLY11]. Während Benutzern das Lösen der Aufgabe dadurch erleichtert wird, kann ein angreifendes Computerprogramm davon nicht profitieren.

Auch Rauschen in Form von Linien und anderen Objekten wird zu dieser Dimension gezählt [YSEA08b]. Die verschiedenen Arten hiervon wurden bereits in Abschnitt 3.2 beschrieben. HIPs mit Rauschen in den verschiedensten Formen konnten bereits erfolgreich gebrochen werden, deshalb ist fraglich ob damit überhaupt Sicherheit gewonnen werden kann. Soll Rauschen einen Angriff abwehren oder zumindest erschweren, so beeinträchtigt es zugleich die Gebrauchstauglichkeit. Gerade wenn Rauschen

sich zwischen Zeichen befindet, erschwert es die Lesbarkeit erheblich. Daraus resultiert, dass Rauschen bestenfalls nicht eingesetzt wird.

4.2.3 Darstellung

Bei textbasierten HIPs fallen unter diese Dimension die verwendete Schriftart und -größe sowie die zum Einsatz kommenden Farben [YSEA08b].

Während Menschen eine wechselnde Schriftart in einem HIP leicht entziffern können, stellt dies für die automatische Texterkennung ein großes Problem dar [Pon06, Sut12]. Gelingt es, ein HIP in die jeweiligen Zeichen zu segmentieren, sollte dies jedoch kein Problem darstellen, da die Texterkennung für jedes Zeichen separat durchgeführt werden kann. Wenn die unterschiedlichen Schriftarten nicht dafür eingesetzt werden eine Segmentierung zu verhindern, kann aus Gründen der Gebrauchstauglichkeit darauf verzichtet werden.

Die Verwendung von Farben muss sehr genau überlegt sein [YSEA08b]. Dabei spielt das Verhältnis der verwendeten Farben die größte Rolle, denn völlig unterschiedliche Farben zu verwenden erhöht die Sicherheit nicht. In Abbildung 4.2 ist hierfür ein Beispiel gezeigt: Da alle Zeichen dieselbe Farbe besitzen, können diese mit einem Schwellwertfilter extrahiert werden [YSEA08b]. Bei der Verwendung von Farben sollten deshalb ähnliche Farben für den Hintergrund und die Zeichen verwendet werden. Diese Farben müssen jedoch sehr sorgfältig gewählt werden, da ein solches HIP möglicherweise zu wenig Kontrast aufweist und nicht von allen Benutzern gut entziffert werden kann. Wird jedes Zeichen in einer anderen Farbe geschrieben, erschwert dies sicher das extrahieren der Zeichen, kann es aber nicht verhindern. Gleiches gilt wenn die Zeichen etwas heller oder dunkler als der Hintergrund gefärbt werden, wie von Guo et al. vorgeschlagen [GWLY11]. Da das HIP einem Angreifer als Bilddaten vorliegt, kann dieser die kleinsten Unterschiede der Farbe in einem Schritt sichtbar machen. Der menschliche Benutzer kann dies nicht und ist auf sichtbare Unterschiede angewiesen. Da der Benutzer hier zweifelsohne benachteiligt ist, sollten nur zwei Farben und kein auffälliger Hintergrund verwendet werden [Pon06, YSEA07, YSEA08b].

Abbildung 4.2: Die bei diesem HIP verwendeten Farben erhöhen die Sicherheit nicht, da sie einfach herausgefiltert werden können (aus [HHY12]).

4.3 Audiobasierte HIPs

Audiobasierte HIPs werden derzeit nur in wenigen Fällen wirklich genutzt. So waren nur 0,77 Prozent der HIPs, die eBay im Jahr 2009 auslieferte, audiobasiert [BBF+10]. Ob dies der geringen Benutzerfreundlichkeit geschuldet ist, konnte jedoch nicht zweifelsfrei festgestellt werden. Die Gestaltungsdimensionen der textbasierten HIPs lassen sich auch auf die Audiobasierten anwenden [YSEA08b]. Diese werden in den folgenden Abschnitten erörtert und diskutiert.

4.3.1 Verzerrung

Im Fall von audiobasierten HIPs wird die Verzerrung durch den Einsatz von Rauschen und Hintergrundgeräuschen erreicht, was die Verständlichkeit des Inhalts sehr einschränkt und deshalb nur in geringem Maße verwendet werden sollte [YSEA08b]. Jedoch wären solche HIPs zu unsicher, weshalb die Audiosamples oft sehr stark verzerrt werden. Dadurch stimmten beispielsweise die Antworten der Testpersonen auf Audio-HIPs von Google und Microsoft in einer Studie nur zu 1 Prozent überein, was jedoch nicht bedeutet, dass Antworten überhaupt korrekt waren [BBF+10]. Dies zeigt anschaulich, wie schwer verständlich die verwendeten Audiosamples sind.

Der Einsatz von *semantic noise* kann das Auffinden der relevanten Teile des Audiosamples erschweren, was einen Angriff möglicherweise unterbindet. Jedoch muss auch in diesem Fall das gesamte Audiosample verzerrt werden, da sonst beispielsweise nur die unverzerrten Teile betrachtet werden müssten. Es ist deshalb fraglich ob die Verständlichkeit dieser HIPs durch *semantic noise* verbessert, oder lediglich mehr Sicherheit gewonnen werden kann.

4.3.2 Inhalt

Auch im Fall von audiobasierten HIPs ist der Inhalt ausschlaggebend. So kann davon ausgegangen werden, dass auch hier mit einer steigenden Menge von unterschiedlichen Inhalten die Sicherheit erhöht werden kann, jedoch bei der Verständlichkeit Abstriche gemacht werden müssen. Die meisten bekannten audiobasierten HIPs verwenden Samples von gesprochenen Zahlen, Buchstaben oder beidem.

Werden hingegen nicht nur einzelne Zeichen gesprochen, sondern ganze Sätze oder Worte, könnte möglicherweise etwas weniger Rauschen verwendet werden. Dabei würde der Sicherheit zugutekommen, dass ein wesentlich komplexeres Audiosignal identifiziert werden müsste. Da Menschen dies ohnehin gewohnt sind, sollte es ohne besondere Anstrengung bewerkstelligt werden können. Allerdings tritt hierbei wieder das Problem der Sprachbarriere auf, denn während bei textbasierten HIPs auch unbekannte Wörter entziffert werden können, ist dies bei audiobasierten HIPs nahezu unmöglich [BBF+10, YSEA08b].

Eine Alternative zu einzelnen Zeichen oder Wörtern testete reCAPTCHA [reC04] im Jahr 2009. Dabei wurden Ausschnitte aus Radiosendungen abgespielt, deren Inhalt erfasst und anschließend grob wiedergegeben werden musste [HCR09]. Dies scheint jedoch nicht erfolgreich gewesen zu sein, da schon 2010 wieder einzelne Zeichen verwendet wurden [BBF+10].

Der zeitliche Aufwand für das Lösen eines solchen HIPs ist enorm. Während sehende Benutzer diese HIPs in durchschnittlich 23 Sekunden lösen können, benötigen Blinde zwischen 30 und 90 Sekunden [BC09, SHFL08]. Der Durchschnitt liegt hier bei 51 Sekunden und ist damit mehr als doppelt so lang. Im Vergleich hierzu können textbasierte HIPs in weniger als zehn Sekunden korrekt gelöst werden [BC09]. Die hohe Zeitdauer liegt neben Verständlichkeitsproblemen auch daran, dass die Audiosamples meist über zehn, manchmal sogar 37 Sekunden lang sind [BBF+10]. Es ist also gar nicht möglich das HIP schneller zu Lösen. Ein zu kurzes Audiosample enthält möglicherweise zu wenig Informationen um eine ausreichende Sicherheit zu bieten, während ein zu langes Sample vor allem bei wiederholter Wiedergabe zu Frust führen kann.

4.3.3 Darstellung

Sichtbare HIPs können im Ganzen betrachtet und erfasst werden. Wenn der Benutzer die Antwortbox fokussiert hat, kann er immer noch einen Blick auf die Aufgabe werfen und die Antwort gegebenenfalls korrigieren [BC09]. Audio-HIPs sind dagegen linear. Ein Benutzer startet die Wiedergabe und fokussiert dann schnell die Antwortbox um die Lösung einzugeben. Für sehende Benutzer bedeutet dies nur einen Klick mit der Maus während blinde Benutzer mithilfe der Tastatur dorthin navigieren müssen. Meistens wird als Unterstützung bei der Navigation ein Screen Reader verwendet, der das gerade ausgewählte Objekt verbal beschreibt. Durch diese Audioausgabe wird in den allermeisten Fällen das HIP gestört und kann nicht korrekt gelöst werden. Um ein Audio-HIP nochmals zu hören, muss zum Start-Button zurücknavigiert und das Audiosample komplett neu gehört werden, was zu sehr langen Bearbeitungszeiten führt [BC09].

Dies legt die Vermutung nahe, dass die derzeitigen Interfaces von Audio-HIPs nicht für blinde Benutzer entworfen wurden [BC09]. Das Interface könnte jedoch angepasst werden, da hiervon die Sicherheit der HIPs nicht negativ beeinflusst wird. Eine einfache Maßnahme ist ein verspätetes Abspielen des Audiosamples und die Möglichkeit die Wiedergabe zu Steuern. So hätten blinde Benutzer Zeit, um zur Antwortbox zu navigieren bevor die Wiedergabe beginnt und würde etwas nicht verstanden, könnte zurückgespult werden um nicht das ganze Sample erneut anhören zu müssen [BC09]. Bei einem Test mit einem so gestalteten HIP konnten 69 Prozent der Audiorätsel beim ersten Versuch korrekt gelöst werden. Dem gegenüber stehen 43 Prozent, die bei einem Standard-HIP erreicht werden konnten [BC09]. Viele Anbieter von Audio-HIPs verwenden Techniken die verhindern, dass die Audiosamples gespeichert werden. Dies ist zwar positiv für die Sicherheit, jedoch wird so auch verhindert, dass blinde Benutzer die Audiosamples mit einem anderen Wiedergabeprogramm, an das sie gewohnt und im Umgang damit sicherer sind, abspielen können [BC09].

Um die Schwierigkeiten der Standard-Interfaces zu umgehen, versuchen viele blinde Benutzer sich die Antwort zu merken und navigieren erst nach Ende der Audiowiedergabe zu der Antwortbox [SHFL08]. Andere verwenden Hilfsmittel wie Diktiergeräte und Texteditoren oder bitten sogar sehende Personen um Hilfe, und sind damit signifikant erfolgreicher als andere Blinde [BC09, SHFL08]. Leider können selbst mit Hilfsmitteln nicht mehr als die Hälfte der HIPs korrekt gelöst werden [BC09, SHFL08].

4.4 Mobiler Einsatz

Herkömmliche HIPs, wie text-, audio- und bildbasierte, sind auf mobilen Endgeräten eher schlecht zu verwenden [MSG+13]. So zoomt Asirra (siehe Abschnitt 2.3) beispielsweise in die angezeigten Bilder wenn darauf gedrückt wird. Dabei verdecken sich die Bilder gegenseitig und der Benutzer muss durch Verschieben der Ansicht gegensteuern [RC13]. Bewegt sich der Benutzer im öffentlichen Raum, sind audiobasierte HIPs wegen Störgeräusche möglicherweise noch schwieriger zu verstehen. Zusätzlich können die Menschen im näheren Umfeld durch die Audiowiedergabe belästigt werden.

Bei textbasierten HIPs wird in den meisten Fällen durch das Auswählen der Textbox die Bildschirmtastatur angezeigt. Dadurch bleibt oftmals so wenig Platz auf dem Display, dass die Aufgabe nicht mehr angezeigt werden kann [RC13]. Hat sich der Benutzer die Antwort nicht gemerkt, muss er immer wieder zwischen Aufgabe und Antwortbox hin und her wechseln. Werden in einem HIP Buchstaben und Zahlen eingesetzt, muss der Benutzer darüber hinaus die Bildschirmtastatur entsprechend umschalten. Zudem kann eine eingeschaltete Autokorrektur eigentlich korrekte Eingaben verfälschen [RC13].

Eine mögliche Lösung für dieses Problem ist ein HIP, in das bereits eine Tastatur integriert ist [LHBL11]. Je nach Anzahl der Zeichen dieser Tastatur würde die Sicherheit leiden, es könnte jedoch für HIPs eingesetzt werden, die ohnehin eine limitierte Anzahl von Zeichen oder nur Ziffern verwenden [LHBL11]. Im Allgemeinen sind die Schaltflächen der Interfaces oft zu klein für den Einsatz auf Touch-Displays [RC13]. Befindet sich der Benutzer mit seinem Gerät in einer hellen Umgebung, kann das Ablesen des Displays nur schwer möglich sein. In diesem Fall kann ein HIP unter Umständen nicht korrekt gelöst werden, da manche HIPs oft nur schwache Kontraste verwenden [RC13].

Vor allem in Bereich der spielbasierten HIPs ist im mobilen Einsatz darauf zu achten, dass durch grafisch aufwändige Spiele die Akkulaufzeit nicht zu stark reduziert wird. Auch ist die Leistungsfähigkeit der heute verbreiteten Smartphones sehr unterschiedlich, sodass Spiele möglicherweise nicht auf allen Geräten flüssig gespielt werden können. Ist ein Benutzer aufgrund dieses Problems nicht in der Lage das Spiel korrekt zu lösen, so ist dessen Frustration wahrscheinlich noch größer als bei anderen HIP-Techniken. Auch kann die automatische Drehung der Anzeige Probleme bereiten. Wird im entscheidenden Moment die Anzeige gedreht, so verliert der Spieler möglicherweise die Kontrolle und das Spiel. Neben diesen Faktoren ist zusätzlich wichtig, die Datenmenge für die Spielausführung so klein wie möglich zu halten. Dadurch werden sowohl die Dauer als auch die Kosten für die Übertragung der Spieldaten reduziert.

4.5 Abgeleitete Richtlinien für das Erstellen von gebrauchstauglichen HIPs

Generell kann gesagt werden, dass eine Eigenschaft, die das Lösen des HIPs erschwert, sich positiv auf die Sicherheit, aber negativ auf die Gebrauchstauglichkeit auswirkt, und umgekehrt. Daraus resultiert ein Zielkonflikt für dessen Lösung auf beiden Seiten Abstriche gemacht werden müssen. Damit jedoch ein gewisses Maß an Gebrauchstauglichkeit eingehalten wird, wurden in diesem Kapitel Eigenschaften aufgezeigt, die ein HIP auf jeden Fall erfüllen muss. Diese Eigenschaften und die Richtlinien, die sich hieraus ableiten lassen, werden in diesem Abschnitt zusammengefasst.

Umso schneller ein HIP von Menschen gelöst werden kann, desto besser [BBF+10]. Denn je kürzer der Benutzer sich damit beschäftigen muss, umso geringer ist auch sein Frustrationslevel. Darüber hinaus sollte das HIP nicht die größte Anstrengung der Benutzer fordern, denn löst es dieser falsch, verdoppelt sich mindestens die benötigte Zeit.

Im Bereich der text- und audiobasierten HIPs besteht eine Sprachbarriere. Immer wenn Wörter einer bestimmten Sprache verwendet werden, werden die Benutzergruppen benachteiligt, die diese Sprache nicht sprechen [BBF+10]. Dies kann bei textbasierten HIPs dadurch umgangen werden, dass zufällige Zeichen anstelle von Wörtern verwendet werden. In diesem Fall bleibt jedoch das Problem bestehen, dass manche Benutzer die verwendeten Zeichen, wie lateinische, arabische, kyrillische oder chinesische nicht kennen [YSEA08b]. Enthalten andere Arten von HIPs eine Aufgabenbeschreibung oder ebenfalls Wörter, so bestehen diese Probleme auch hier.

In textbasierten HIPs sollten so viele unterschiedliche Zeichen wie möglich verwendet werden [GWLY11]. Als limitierender Faktor wird an dieser Stelle die Unterscheidbarkeit der Zeichen genannt [YSEA08b]. Auch kann die Verwendung von immer gleich langen Zeichenketten nicht empfohlen werden. Abgesehen vom Problem der Sprachbarriere sollten in einem solchen HIP Wörter unterschiedlicher Länge eingesetzt werden, da der Benutzer hierdurch einige Vorteile gewinnt. So muss er nicht jedes Zeichen sicher identifizieren, sondern kann implizit darauf schließen [YSEA07]. Um die Sprachbarriere zu verkleinern können auch phonetisch korrekt klingende Zeichenfolgen verwendet werden,

wie dies von Google praktiziert wird [BBF+10, GWLY11]. Als Wortlänge werden fünf bis maximal zehn Zeichen empfohlen. Kommen anstelle von Wörtern Ketten aus zufällig gewählten Zeichen zum Einsatz, so sollten maximal acht Zeichen verwendet werden.

Die Verzerrung von Zeichen ist an sich unproblematisch, auch die Rotation sowie das Deformieren und Verschieben von Zeichen behindert den Menschen nur wenig [CLSC05a]. Jedoch sind Computer bei der Identifikation dieser so bearbeiteten Zeichen dem Menschen voraus [CLSC05b]. Umso weiter dies getrieben wird, desto größer wird dieser Vorsprung. Deshalb ist es wichtig diese Modifikation mit anderen Techniken zu kombinieren, welche die Segmentierung der HIPs verhindern [CLSC05b, HLBO10]. Durch das enge Setzen der Zeichen wurde bereits ein hohes Maß an Segmentierungssicherheit erreicht. Jedoch entsteht dadurch das Problem, dass Zeichenpaare von anderen Zeichen nicht mehr sicher zu unterscheiden sind [YSEA08b]. Sollte kein anderer Segmentierungsschutz entwickelt werden, ist deshalb die Zukunft der textbasierten HIPs fraglich. Denn während die Texterkennung in Zukunft weiter verbessert werden kann, werden die Fähigkeiten des Menschen eher gleich bleiben.

Während bei audiobasierten HIPs für eine ausreichende Sicherheit Rauschen eingesetzt werden muss (siehe Abschnitt 3.4), sollte bei textbasierten HIPs davon abgesehen werden. Denn wie gezeigt wurde, konnte das Rauschen bei diesen HIPs bisher immer umgangen werden (siehe Abschnitt 3.2). Wenn das Rauschen eine Segmentierung nicht verhindern kann, stört es lediglich die Lesbarkeit für den Menschen, was unbedingt zu vermeiden ist. Auch sollte im textbasierten Bereich keine wechselnde Schriftart verwendet werden. Dies beeinträchtigt ebenfalls die Lesbarkeit während es einem Angriff nach einer erfolgreichen Segmentierung nichts entgegen zu setzen hat. Gleiches gilt für den Einsatz von mehr als zwei unterschiedlichen Farben, da ein Angreifer hier viel bessere Möglichkeiten zur Unterscheidung besitzt als ein normaler Benutzer [Pon06, YSEA07, YSEA08b].

Die meisten der heute verwendeten audiobasierten HIPs bieten Verbesserungspotential, denn sie können nur sehr schlecht verstanden werden [BBF+10]. Darüber hinaus sind sie von blinden Benutzern nicht ohne Hilfsmittel lösbar, weshalb die Interfaces überarbeitet werden sollten [BC09]. Ein verspätetes Starten des Audiosamples und auch nur minimale Eingriffsmöglichkeiten in die Steuerung der Wiedergabe können hier eine große Hilfe sein [BC09].

Auch das Problem der Sprachbarriere ist hierbei noch gravierender [BBF+10, YSEA08b]. Dabei dürfte es keinen Unterschied machen ob einzelne Zeichen oder ganze Wörter bzw. Sätze vorgespielt werden. Jedoch sollte die Alternative wiedergegebener Sätze anstelle von Zeichen weiter untersucht werden, da hierbei aufgrund der gesteigerten Komplexität der Aufgabe die Chance besteht, dass das Audiosample weitaus weniger verzerrt und mit weniger Rauschen hinterlegt werden muss [HCR09].

Im Allgemeinen können audiobasierte HIPs nicht so schnell wie textbasierte gelöst werden, was jedoch für ein geringes Frustrationslevel des Benutzers notwendig wäre [BC09, SHFL08]. Der so benötigte relativ hohe Zeitaufwand wird grundlegend durch die Länge des Audiosamples vorgegeben [BBF+10]. An dieser Stelle sollte überprüft werden, ob dies nicht auch wesentlich kürzer ausfallen könnte als die heute verwendeten.

Für den benutzerfreundlichen Einsatz auf mobilen Endgeräten im Allgemeinen und auf Smartphones im Speziellen müssen die heutigen HIPs weiter angepasst werden. Denn aufgrund der zumeist beschränkten Displaygröße können die HIPs nicht immer komplett angezeigt werden und frustrieren den Benutzer durch ständiges hin und her wechseln [MSG+13, RC13]. Im Bereich der textbasierten HIPs sollte der Einsatz der Bildschirmtastatur überdacht werden. Gerade wenn Zahlen und Buchstaben verwendet werden, muss diese oft umgeschaltet werden. Eine Integration der benötigten Tasten in das HIP könnte eine Lösung hierfür darstellen [LHBL11]. Wird ein spielbasiertes HIP für den mobilen Einsatz

konzipiert, ist darauf zu achten, dass die Akkulaufzeit nicht durch grafisch aufwändige Spiele zu sehr verkürzt wird. Außerdem ist zu überprüfen ob eine automatische Drehung des Displays auftreten und zu Problemen mit der Steuerung führen kann. Auch sollte die Datenmenge, die für die Ausführung des Spiels übertragen werden muss, aufgrund von teuren Volumeneinheiten so klein wie möglich sein.

Der Zielkonflikt zwischen Sicherheit und Gebrauchstauglichkeit wird durch die sprachlichen und kulturellen Barrieren weiter verstärkt. Da diese vor allem bei text- und audiobasierten HIPs auftreten, ist deren benutzerfreundliche Gestaltung besonders anspruchsvoll. Bei der Entwicklung eines neuen HIPs sollte deshalb versucht werden, auf eine Aufgabenstellung zu verzichten. Kommt das HIP ohne eine solche oder anders geartete Anleitung aus, ist es vielen anderen HIPs gegenüber im Vorteil.

Leider konnten aufgrund der geringen Verbreitung von spielbasierten HIPs keine Publikationen zu diesem Thema gefunden werden. Es steht jedoch außer Frage, dass auch für diese HIPs die allgemeinen Anforderungen gelten. So ist auch hier beispielsweise auf eine möglichst kurze Bearbeitungsdauer zu achten. Kommen diese HIPs ohne eine Anleitung aus und bedienen sie sich eines Konzeptes, das in vielen Kulturkreisen bekannt ist, so bieten diese eine sehr gute Alternative zu den bisher verwendeten HIPs. Da spielbasierte HIPs den Benutzer eher unterhalten wollen, als ihn vor eine schwierige Aufgabe zu stellen, ist auch die Akzeptanz in den meisten Fällen größer [Are14a].

5 Studie

In den beiden vorherigen Kapiteln wurde gezeigt, dass die bisher verwendeten HIP Techniken nur bedingt Schutz vor automatisierten Angriffen bieten. Darüber hinaus leidet die Gebrauchstauglichkeit oft unter dem Versuch, ein HIP sicherer zu gestalten. Dies gilt ebenso für den Einsatz der HIPs auf mobilen Endgeräten (siehe hierzu Abschnitt 4.4). Da zu erwarten ist, dass die Nutzung des Internets durch diese Geräte weiter zunimmt (vgl. [D2113, Seite 8]), ist es wichtig speziell für diesen Einsatz neue HIPs zu entwerfen [LHBL11].

Mit dieser Studie soll geprüft werden, inwieweit ein HIP für den Einsatz auf mobilen Endgeräten entworfen werden kann. Des Weiteren soll diese Studie Klarheit über die Sicherheit und Gebrauchstauglichkeit eines solchen neuartigen HIPs bringen und zeigen ob der Zielkonflikt, der sich aus diesen beiden Aspekten ergibt, durch die Verwendung neuer Eingabemedien entschärft werden kann. Dazu werden zuerst die Anforderungen definiert, die an ein solches HIP gestellt werden müssen. Diese Anforderungen wurden aus den beiden vorherigen Kapiteln abgeleitet. Darauf folgt eine Auflistung von möglichen Umsetzungen eines solchen HIPs, welche die genannten Anforderungen erfüllen können. Aus dieser Auflistung wurde eine Umsetzung ausgewählt und implementiert. Eine genaue Beschreibung dieses HIPs folgt in Abschnitt 5.3. In den darauf folgenden Abschnitten werden die Ergebnisse der Evaluation hinsichtlich der Sicherheit und der Gebrauchstauglichkeit beschrieben und mögliche Verbesserungen dargelegt. Am Ende dieses Kapitels findet sich ein Resümee, in dem alle wesentlichen Erkenntnisse, die aus dieser Studie gewonnen werden konnten, noch einmal zusammengefasst sind.

5.1 Anforderungen

In diesem Abschnitt werden diejenigen Anforderungen zusammengefasst, welche das neu entwickelte HIP erfüllen muss. Dabei erfolgt die Auswahl aus den Anforderungen an die Sicherheit und Gebrauchstauglichkeit, die in den beiden vorangegangen Kapiteln identifiziert wurden. Anforderungen, welche sich beispielsweise speziell auf textbasierte HIPs beziehen, sind für dieses HIP nicht von Belang, behalten jedoch dennoch ihre Gültigkeit.

Eine häufig auftretende Schwierigkeit beim Einsatz von HIPs auf mobilen Geräten ist, dass die eingeblendete Bildschirmtastatur die Displaygröße derart verringert, dass das HIP nicht in voller Größe angezeigt werden kann. Eine Lösung für dieses Problem ist die Verwendung alternativer Eingabemöglichkeiten. Heutige mobile Geräte sind mit einer Vielzahl von unterschiedlichen Sensoren ausgestattet. Darunter sind Helligkeits-, Annäherungs-, Bewegungs- sowie Lagesensoren. Aktuelle Smartphones wie das Samsung Galaxy S4 sind zusätzlich noch mit Thermometer, Barometer und einem Sensor zur Gestensteuerung ausgestattet [Chi13]. Durch die Verwendung solcher Sensoren kann der Einsatz der Bildschirmtastatur entfallen und das HIP wird nicht verdeckt. Bei der verbauten Kamera handelt es sich zwar um einen Sensor, doch eignet sich diese nicht für ein spielbasiertes HIP. Möglich wäre beispielsweise den Benutzer aufzufordern, ein Bild von drei ausgestreckten Fingern zu machen und dieses hochzuladen. Jedoch stellt diese Aufgabe eine Prüfungssituation dar, und das Verstehen der Aufgabe erfordert eine kognitive Leistung. Darüber hinaus würde durch die Aufgabenstellung eine Sprachbarriere entstehen. Es müssen also besser geeignete Sensoren verwendet werden.

Das neu entwickelte HIP muss im Browser dargestellt werden können. Auf eine eigenständige App wird deshalb verzichtet, da diese zuerst auf dem Gerät des Nutzers installiert werden müsste. Zudem erscheint der Einsatz eines HIPs in einer App fragwürdig, da sich in einem solchen Fall der Benutzer bereits mittels der App authentifiziert hat. Der Browser stellt auf der anderen Seite einen limitierenden Faktor dar, da nicht auf alle Daten und Schnittstellen des Betriebssystems des mobilen Endgeräts zugegriffen werden kann. Eine größere Menge von Sensoren könnte möglicherweise durch zusätzliche Software abrufbar sein. So bestünde die Möglichkeit Daten anderer Sensoren durch die Verwendung von Frameworks wie PhoneGap[1] abzufragen, jedoch gilt dies wiederum nicht für alle Sensoren oder die Geräte aller Hersteller [Pho]. Deshalb wird bei diesem HIP auf deren Einsatz verzichtet.

Auch auf Frameworks, die zur Implementation des HIPs verwendet werden könnten, wird aufgrund von Kompatibilitätsproblemen verzichtet. So sind beispielsweise Flash-Anwendungen auf vielen Smartphones nicht lauffähig. Auch das Ausführen von Java-Code im Browser wird auf mobilen Endgeräten nicht unterstützt. Browser Add-Ons könnten ebenfalls die Entwicklung vereinfachen, sind jedoch für mobile Browser nicht verfügbar. Aus diesen Gründen wird für die Entwicklung auf Seite des Clients nur JavaScript zum Einsatz kommen. Die Programmiersprache auf der Seite des Servers unterliegt nur der Einschränkung, dass damit HTML-Seiten generiert werden müssen. Für diese Studie wurde PHP ausgewählt, da es sich dabei um die am weitesten verbreitete Serverskriptsprache handelt [Sur]. Zudem kann diese Sprache kostenfrei verwendet werden, weshalb sie auf vielen Webservern vorinstalliert ist.

Da bereits gezeigt wurde, dass die Akzeptanz für spielbasierte HIPs größer ist als für andere Techniken [Are14a], wird auch in dieser Studie ein spielbasiertes HIP erprobt. Ferner wird darauf geachtet, dass es in kurzer Zeit lösbar ist und dadurch zu weniger Frustration beim Benutzer führt. Auch kann ein kurzes Zeitlimit die Sicherheit erhöhen, da ein Angriff hierfür sehr effektiv sein muss [MSG+13]. Um die Kriterien Sicherheit und Gebrauchstauglichkeit zu erfüllen, sollte es den Benutzern des neuen HIPs möglich sein, es in mehr als 80 Prozent der Versuche korrekt zu lösen. Wird das HIP jedoch von einem Computer gelöst, darf diese Quote nicht mehr als 0,01 Prozent betragen.

Gerade bei spielbasierten HIPs kann eine weitere Limitation im mobilen Bereich die eingeschränkte Rechen- bzw. Grafikleistung der Geräte sein. Darüber hinaus wird durch rechenintensive Anwendungen die Akkulaufzeit verringert. Auch sollte die benötigte Datenmenge, die durch aufwändige Spielgrafiken entstehen könnte, nicht außer Acht gelassen werden, da diese vor allem im mobilen Bereich zu hohen Kosten des Nutzers führen kann. Darüber hinaus sollte überprüft werden, ob die automatische Drehung des Displays zu Problemen führt. Zudem benötigen spielbasierte HIPs eine sehr große Datenbank, um eine möglichst große Anzahl von unterschiedlichen Spielen bieten zu können [MSG+13]. Diese Anforderung kann jedoch auch durch eine zufällige Generierung des Spiels erfüllt werden.

Das HIP sollte überdies keine Sprachbarriere aufbauen, was bei anderen HIP-Techniken unumgänglich ist. Ebenso sollten keine Kulturkreise ausgeschlossen werden. Es muss demnach ein Spiel entwickelt werden, welches ohne Anleitung zu verstehen ist und zudem nicht auf kulturspezifischen mentalen Modellen beruht. Neben diesen Anforderungen sollte das HIP den Benutzer nicht wie bisher vor eine Prüfungssituation stellen. Statt einer schwierigen Aufgabe, wie das Entziffern von Text, soll der Benutzer vielmehr durch das HIP selbst animiert werden, es zu lösen.

Einem Angreifer sollte nicht unmittelbar ersichtlich sein, ob der Angriff erfolgreich ist. Aus diesem Grund sollte das HIP dies verschleiern, oder zumindest die Möglichkeit hierzu bieten. Wie dies realisiert werden kann, wurde bereits in Abschnitt 2.6 beschrieben.

[1] http://phonegap.com/

Die Gebrauchstauglichkeit dieses HIPs sollte nach Abschluss der Entwicklung durch einen breiten Personenkreis getestet werden. Für diese Studie bietet sich die Form einer Onlineumfrage an, da die Testpersonen ohnehin ihr eigenes Gerät verwenden sollen, da sie den Umgang damit gewohnt sind. Außerdem kann so die Kompatibilität mit mehreren verschiedenen Geräten getestet werden. Darüber hinaus erscheint der Test an einem zentralen Ort zu aufwändig.

5.2 Auswahl der Umsetzung

Derzeit sind ohne Hilfsmittel im Browser nur die Daten von zwei verschiedenen Sensoren verfügbar: Die Beschleunigungs- und Lagesensoren. Gegenüber den Beschleunigungssensoren, welche die Bewegung des Geräts im Raum messen, wandeln die Lagesensoren ausschließlich die Rotation des Geräts um seine drei Achsen in Signale um. Abbildung 5.1 zeigt diese drei Achsen sowie die Bezeichnung, die im weiteren Verlauf verwendet wird. Da mit den Beschleunigungssensoren beispielsweise nur das Schütteln des Geräts erkannt werden kann, eignen sich diese Sensordaten nicht für ein Spiel mit komplexen Bewegungsabläufen. Diese sind jedoch wichtig, um eindeutig auf einen menschlichen Benutzer schließen zu können, denn ein Schütteln könnte vergleichsweise einfach von einer Vorrichtung imitiert werden. Deshalb wurden die Lagesensoren, auch Gyroskop genannt, als Eingabemedium für das HIP ausgewählt. Die durch das Gyroskop gemessenen Daten sind durch den derzeit aktuellen HTML5 Standard via JavaScript im Browser abrufbar [W3C]. Da sich diese Sensoren durch eine sehr geringe Stromaufnahme auszeichnen und die Akkulaufzeit der mobilen Endgeräte nicht stark verkürzen [KMD13], können sie ohne Bedenken für ein HIP verwendet werden. Besonders wenn das Lösen des HIPs nur sehr kurze Zeit in Anspruch nimmt.

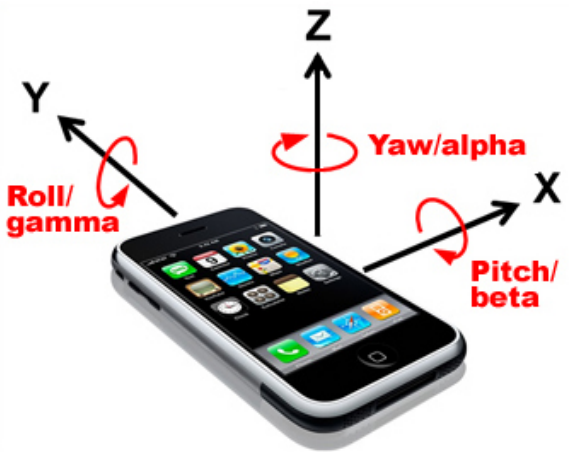


Abbildung 5.1: Für dieses HIP wurden die Lagesensoren verwendet, welche die Rotation um die drei Achsen, und nicht die Bewegung des Geräts im Raum messen (aus [Tha13]).

Ändert sich die Ausrichtung des Geräts, wird im Browser das *DeviceOrientationEvent* ausgelöst. Dieses Event liefert die Drehung um die Achsen *alpha*, *beta* und *gamma* durch gleichnamige Attribute, die im JavaScript-Code der geladenen Seite verwendet werden können. Die Bewegung um die Z-Achse (*alpha*) wird für dieses HIP nicht verwendet. Während *beta* die Drehung von -180 bis +180 Grad angibt, liefert *gamma* nur Werte zwischen -90 und +90 Grad, diese Extreme sollten jedoch für dieses HIP nicht von Belang sein, da hier nur geringe Lageänderungen gefordert sind.

Diese Sensordaten können für die unterschiedlichsten Spiele verwendet werden, die folgende Auflistung erhebt deshalb keinen Anspruch auf Vollständigkeit. All diese Spiele eignen sich jedoch für folgende

Verwendung der Sensordaten: Die Daten der Sensoren werden gelesen und für die Steuerung des Spiels verwendet. Gleichzeitig wird durch eine Auswertung dieser Daten entschieden, ob es sich bei dem Spieler um einen Menschen oder eine Maschine handelt.

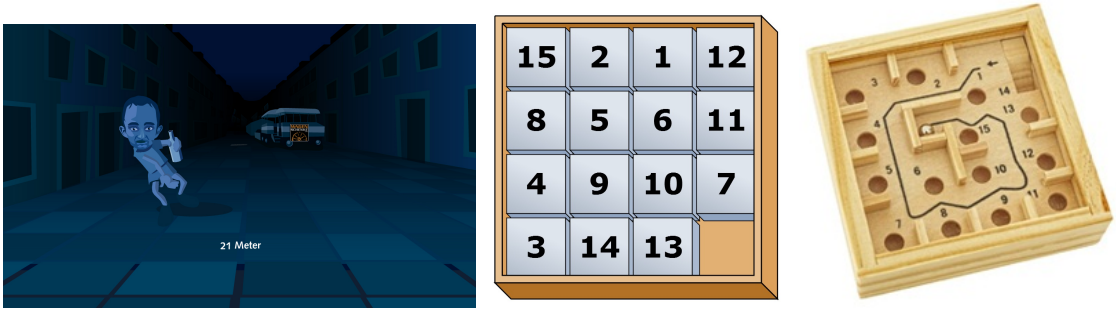
- **Objekt in Bereich halten:** Durch Neigen des Geräts muss der Spieler beispielsweise einen baufälligen Turm in der Waage halten. Gelingt dies für einen relativ kurzen Zeitraum, so ist das Spiel gewonnen. Eine andere mögliche Ausführung dieses Konzeptes ist in Abbildung 5.2(a) mit einer Momentaufnahme aus dem Spiel HomeRun gezeigt. Dort muss der Spieler das Gleichgewicht eines Betrunkenen halten.

- **Schiebepuzzle:** Ein solches Puzzle ist in Abbildung 5.2(b) zu sehen. Durch die fehlende Kachel können die anderen hin und her geschoben werden, bis schließlich die Zahlenreihe oder das Bild korrekt zu sehen ist. Die Steuerung der Verschiebung könnte ebenfalls durch die Lagesensoren des Geräts erfolgen.

- **Kugellabyrinth:** Bei diesem Spiel wird durch die Neigung des Geräts eine Kugel gesteuert. Das Ziel ist es, die Kugel in einen definierten Bereich rollen zu lassen. Dies wird jedoch durch ein Labyrinth und zahlreiche Löcher erschwert, wie Abbildung 5.2(c) zeigt.

- **Torwart:** Die Aufgabe für den Spieler könnte sein, den Torwart durch Neigen des Geräts in die richtige Richtung fallen zu lassen. Gelingt ihm das rechtzeitig, um den Ball zu stoppen bevor dieser ins Tor gelangt, hat der Spieler gewonnen.

- **Rennspiel:** Dieses Genre bietet eine Vielzahl von verschiedenen Ausführungen. So kann beispielsweise ein Auto, Flugzeug, Pferd oder Boot gesteuert werden. Für ein solches Spiel kann das Gerät quasi als Lenkrad fungieren und dessen Neigung das Gefährt steuern.



(a) Bei dem Spiel HomeRun [Wag] muss der Spieler das Gleichgewicht eines Betrunkenen halten.

(b) Bei einem Schiebepuzzle müssen die Kacheln in die korrekte Position geschoben werden (aus [Wik]).

(c) Bei einem Kugellabyrinth muss die Kugel um die Löcher herum gerollt werden (aus [Pre]).

Abbildung 5.2: Beispiele für die mögliche Umsetzung eines Browserspiels für diese Studie.

Aus diesen möglichen Umsetzungen wurde das Kugellabyrinth ausgewählt, da hierfür eine sehr empfindliche Steuerung notwendig ist und somit erstens überprüft werden kann, ob derart empfindliche Sensoren in allen gängigen Geräten verbaut sind. Zweitens ermöglichen kleine Änderungen um eine neutrale Ausrichtung des Geräts möglicherweise tiefergehende Analysemöglichkeiten gegenüber einem Vollausschlag nach links oder rechts, wie es beispielsweise bei dem Torwart- oder Rennspiel ausreichen würde. Außerdem stellt dieses Spiel relativ geringe Anforderungen an die Grafikleistung der Geräte, da hier die Kugel das einzige sich bewegende Objekt ist.

5.3 Beschreibung des umgesetzten HIPs

Für diese Studie wurde ein spielbasiertes HIP in Form eines Kugellabyrinths implementiert. Dieses HIP wird an dieser Stelle genauer beschrieben und in den folgenden Abschnitten 5.4 und 5.5 evaluiert. Da es sich bei einem Kugellabyrinth in der Regel um ein Geschicklichkeitsspiel handelt, und so möglicherweise von vielen Benutzern nur bedingt oder nach mehreren Fehlversuchen gelöst werden kann, wurde das Spiel in dieser Umsetzung vereinfacht. So wurden die Fallen des Labyrinths entfernt und stattdessen die Zielfläche in Form eines Lochs implementiert. Darüber hinaus wurde das Labyrinth sehr vereinfacht und stellt nun keine kognitive Herausforderung mehr dar, da der Weg zum Ziel bereits auf den ersten Blick zu sehen ist.

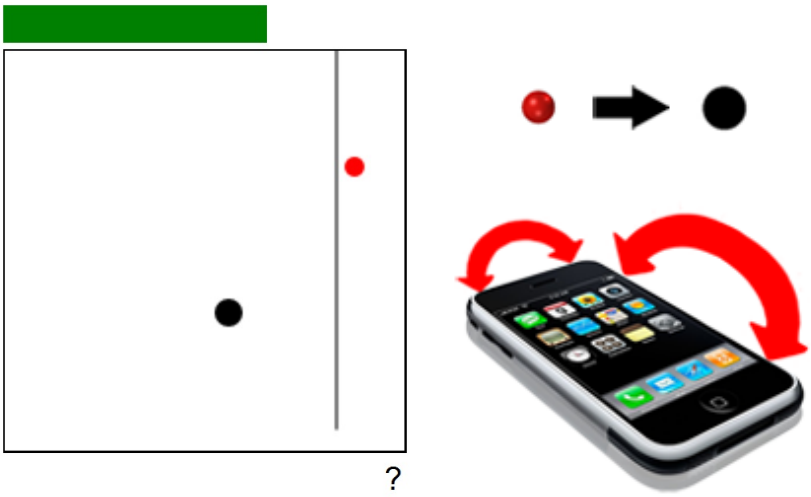
Die Lageänderung des Geräts wird bei diesem Spiel für die Kontrolle einer virtuellen Kugel verwendet. Das Ziel des Spiels ist es, die Kugel innerhalb des Zeitlimits durch die entsprechende Neigung des Geräts in ein Loch rollen zu lassen. Hierbei ist zu beachten, dass eine Rotation um die Y-Achse (*gamma*) eine Bewegung der Kugel entlang der X-Achse verursacht. Gleiches gilt für die Rotation um die X-Achse (*beta*), welche die Kugel entlang der Y-Achse rollen lässt. Alle in diesem Kapitel folgende Bezeichnungen der X- und Y-Achsen beziehen sich auf die Bewegung der Kugel, während die Rotationsrichtung mit *beta* und *gamma* angegeben wird.

Abbildung 5.3(a) zeigt eine Momentaufnahme des Spiels, für welches der Benutzer zehn Sekunden Zeit hat. Nach deren Ablauf wird das Spiel automatisch neu gestartet, wobei die Kugel, das Loch und die Barriere neu positioniert werden. Dazu werden zufällig Koordinaten für die Kugel und das Loch generiert, wobei in X-Richtung stets ein Mindestabstand eingehalten wird. Auch wird das Loch nie am Rand platziert, da die Kugel sonst einfacher eingelocht werden könnte. Zwischen Kugel und Loch wird darüber hinaus in zufälligem Abstand eine Barriere gezogen, die entweder am oberen oder unteren Rand eine kleine Öffnung aufweist, durch welche die Kugel gerollt werden muss. Lediglich die Größe der gesamten Spielfläche sowie die der Kugel und des Lochs sind fest vorgegeben und ändern sich nie. Diese Abmessungen wurden während der Entwicklung auf diesen Wert festgelegt und sind in Abschnitt 5.3.2 näher beschrieben. Stößt die Kugel an den Rand der Spielfläche oder die Barriere, so prallt sie von dort zurück. Wird das Gerät gerade gehalten, verringert sich die Geschwindigkeit der Kugel langsam, bis sie zum Stillstand kommt.

Um keine Sprachbarriere aufzubauen, wurde in dem gesamten HIP kein Text verwendet. Stattdessen ist die Anleitung in Piktogrammen verfasst und die Zeit wird in Form eines Balkens angezeigt, der mit fortschreitender Zeit immer kürzer wird und zudem die Farbe ändert. So wird dieser Balken nach der Hälfte der Zeit Gelb und kurz vor Ablauf der Zeit schließlich Rot. Es wurden diese Farben gewählt, da diese den weltweit verwendeten Ampelfarben entsprechen und somit leicht zu verstehen sind. Die Anleitung kann durch Drücken auf ein Fragezeichen unterhalb der Spielfläche aufgerufen werden und ist in Abbildung 5.3(b) gezeigt.

Darüber hinaus sollte dieses HIP frei von kulturellen Barrieren sein. So wurden anstelle von komplexen Strukturen oder Abbildungen allein zwei Kreise verwendet. Da bei diesem Spiel einzig die physikalischen Eigenschaften der Gravitation und Massenträgheit zum Tragen kommen, kann davon ausgegangen werden, dass das Konzept in allen Kulturkreisen verständlich ist oder innerhalb kürzester Zeit begriffen werden kann.

Da bei diesem Spiel die Kugel innerhalb der vorgegebenen Zeit in das Loch gerollt werden muss, gestaltet sich das verschleiern des Ergebnisses sehr schwer, was jedoch aus Gründen der Sicherheit gefordert ist (siehe Abschnitt 2.6). So ist einem Angreifer einerseits klar, dass er das Spiel erfolgreich



(a) Momentaufnahme des HIPs, das für diese Studie entwickelt wurde. Die rote Kugel muss durch die entsprechende Neigung des Geräts in das Loch gerollt werden.

(b) Die Anleitung des neu entwickelten HIPs, die durch drücken auf ein Fragezeichen unterhalb der Spielfläche geöffnet werden kann.

Abbildung 5.3: Verschiedene Screenshots, die Teile des neu entwickelten HIPs zeigen.

gelöst hat, wenn die Kugel in das Loch gelangt ist. Andererseits erfolgt die Auswertung der gesammelten Daten erst im Anschluss daran. Dieses Verfahren orientiert sich an den spielbasierten HIPs, welche in Abschnitt 2.4 beschrieben wurden. Das Ergebnis der Auswertung kann in Abhängigkeit des Verwendungszwecks[2] verschleiert werden. Da es sich hierbei nur um eine Studie handelt, wurde jedoch darauf verzichtet, da die direkte Rückmeldung für Testzwecke sehr nützlich sein kann.

Sollte ein Gerät aufgrund fehlender Sensoren oder veralteter Software (es wird ein HTML5 tauglicher Browser benötigt) nicht zu diesem Spiel kompatibel sein, besteht die Möglichkeit ein alternatives bzw. herkömmliches HIP anzuzeigen. Jedoch war dies für diese Studie nicht essentiell und wurde deswegen nicht implementiert. Stattdessen wird nur eine entsprechende Fehlermeldung angezeigt. Die aufgetretenen Kompatibilitätsprobleme werden in Abschnitt 5.5.2 näher beschrieben.

5.3.1 Ablauf des Spiels

Abbildung 5.4 zeigt den Ablauf der Kommunikation zwischen dem mobilen Endgerät des Benutzers und dem Server, der das HIP zur Verfügung stellt. Mit der ersten Anfrage des Geräts erstellt der Server eine neue Session und generiert ein neues Spiel. Dazu gehören zum einen die Koordinaten der Spielelemente wie Kugel, Loch und Barriere, und zum anderen auch ein Zeitstempel. Diese Daten werden serverseitig in der Session gespeichert und das Spiel an das Gerät des Benutzers ausgeliefert.

Gelingt es dem Spieler nicht innerhalb von zehn Sekunden die Kugel einzulochen, wird das Spiel neu gestartet. Dazu wird die Seite automatisch neu geladen, was wiederrum ein komplett neues Spiel erzeugt. Die Session bleibt dabei erhalten. Die Daten des Lagesensors werden während des Spiels für die Steuerung der Kugel verwendet und zudem temporär gespeichert. Nur wenn die Kugel innerhalb des Zeitlimits eingelocht wird, werden diese gespeicherten Daten zum Server übertragen. So wird

[2] Beispielsweise im Umfeld eines Kontaktformulars (siehe Abschnitt 2.6).

verhindert, dass Daten unnötig übertragen werden, was im mobilen Bereich unbedingt zu vermeiden ist. Der Server wertet die erfassten Daten aus und gibt dem Benutzer entsprechend Rückmeldung. In diesem Stadium handelt es hierbei nur um eine Ausgabe auf dem Display, während dies später durch eine Weiterleitung zu dem gewünschten Webservice ersetzt werden könnte.

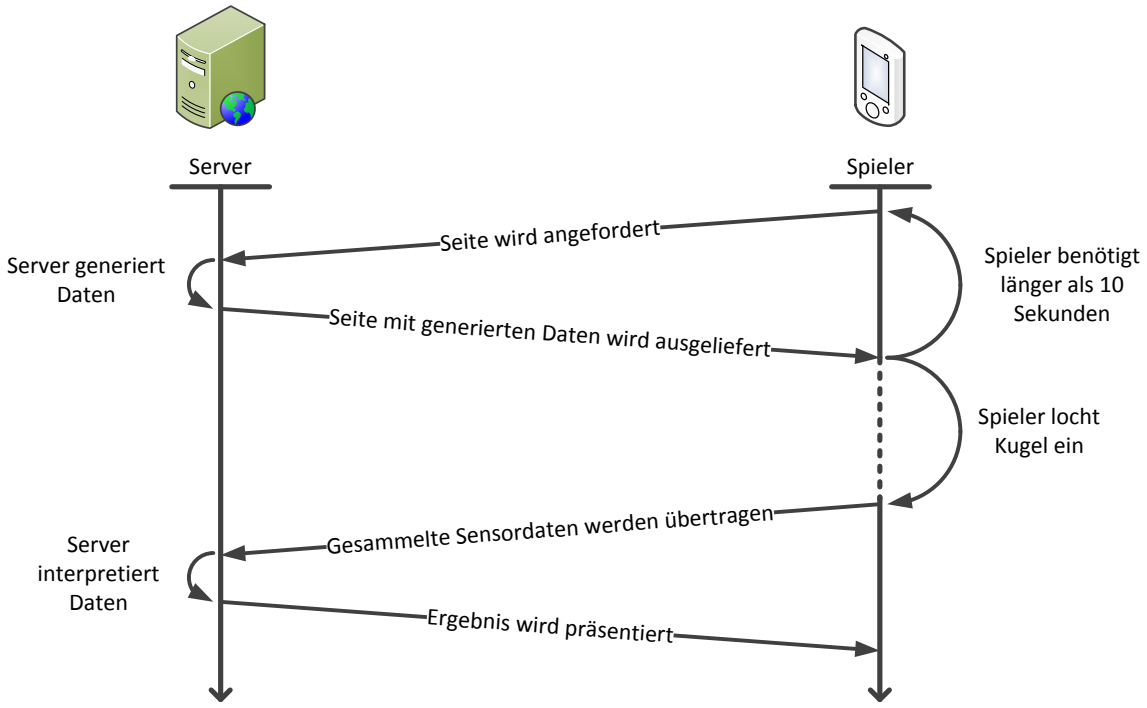


Abbildung 5.4: Schematischer Ablauf der Kommunikation zwischen dem mobilen Endgerät des Spielers und dem HIP-Server.

Wie alle HIPs basiert auch dieses auf der Client-Server-Architektur. Hierbei generiert der Server das Spiel und wertet anschließend die Sensordaten aus. Auf diese Weise erfolgt die Kommunikation nur zu Beginn und am Ende eines jeden Spiels, was das Datenvolumen gering hält und auch unempfindlicher gegenüber hohen Latenzen ist.

Der Quellcode des Spiels wird an dieser Stelle kurz beschrieben und ist im Anhang zu finden. Die PHP-Dateien werden auf dem Server ausgeführt, welcher mittels der *index.php* ein Spiel mit zufälligen Koordinaten generiert, das anschließend an den Client gesendet wird. Nach dem Spiel werden die gesammelten Daten an die Datei *check.php* (siehe Anhang 7.4) gesendet. Darauf folgt ein Aufruf der *result.php* (Anhang 7.5), die das Ergebnis der Datenprüfung darstellt. Die Aufrufe der Dateien sind im zeitlichen Verlauf in Abbildung 5.5 zu sehen. Aus diesem Netz-Log ist ebenfalls die sehr geringe Datenmenge ersichtlich, die zwischen Client und Server ausgetauscht wird. In diesem Beispiel beträgt diese lediglich 9 kB.

Durch das Generieren des Spiels auf dem Server kann dieser am Ende mit größerer Sicherheit überprüfen ob das Spiel erfolgreich gelöst wurde. Denn würde das Spiel im Browser des mobilen Geräts generiert werden, hätte das HIP-System keine Kontrolle darüber, ob wirklich ein neues Spiel gespielt wurde oder immer wieder dieselben Daten verwendet werden. Um dies ausschließen zu können, müssten unzählige Spiele serverseitig gespeichert werden. Darüber hinaus ist es so möglich, dem Angreifer ein zeitliches Limit vorzugeben. Generiert der Server das Spiel, so speichert er zusätzlich einen Zeitstempel ab. Löst der Benutzer das Spiel, wird der Zeitstempel der Spielerstellung mit der aktuellen Zeit verglichen. Liegt

hier eine zu große Differenz vor, wird das Spiel als nicht erfolgreich bewertet. Andernfalls wäre es einem Angreifer wiederum möglich, die Daten so zu manipulieren, dass diese den Eindruck erwecken, das Spiel wäre innerhalb der vorgegebenen Zeit gelöst und die Daten nur zu einem späteren Zeitpunkt abgesendet worden. Zusätzlich zu den gesammelten Sensordaten werden die Startkoordinaten der Kugel sowie die des Lochs übertragen und vom Server abgeglichen. Dadurch soll sichergestellt werden, dass die empfangenen Sensordaten auch zu dem Spiel dieser Session gehören. Andernfalls wäre es möglich, valide Daten für mehrere Spiele zu verwenden.

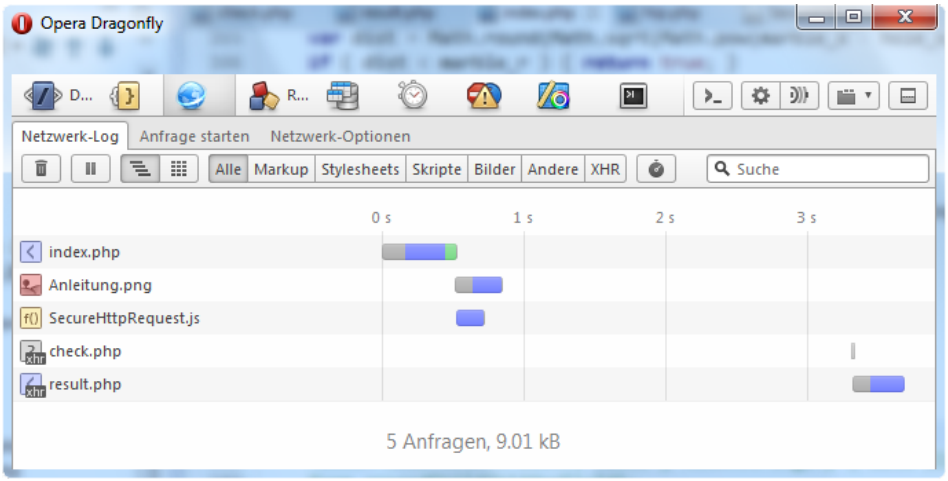


Abbildung 5.5: Analyse der Seitenaufrufe des HIPs während eines Spiels, inklusive der Anzeige des Ergebnisses.

Die Spielfläche wird in einem Intervall von 40 ms neu gezeichnet. Dazu wird aus den, zu diesem Zeitpunkt vorliegenden Sensordaten die Geschwindigkeit der Kugel berechnet und die Kugel dann an der neuen Position gezeichnet. Alle anderen Elemente verbleiben an der gleichen Stelle. Anschließend werden die verwendeten Sensordaten zusammen mit der aktuellen Bildnummer gespeichert. Als Kompromiss zwischen Genauigkeit und Datenvolumen werden nur die Sensordaten gespeichert, die auf das Spiel Einfluss nehmen. Denn durch die sehr empfindlichen Sensoren kann es vorkommen, dass sich die gemessenen Sensordaten mehrmals innerhalb weniger Millisekunden ändern. Es wird jedoch nur alle 40 ms die Geschwindigkeit der Kugel berechnet und die Spielfläche neu gezeichnet. Dementsprechend werden nur die zu diesem Zeitpunkt aktuellen Messwerte der Sensoren gespeichert. Um die Datenmenge weiter zu reduzieren, werden die Sensordaten nur gespeichert wenn diese sich zum vorausgegangen Zyklus geändert haben.

Nach dem erfolgreichen Abschluss des Spiels werden die gesammelten Daten an den Server übertragen und dort analysiert. An dieser Stelle werden folgende Überprüfungen durchgeführt:

- Die vom Spieler gesendeten Startkoordinaten der Spielelemente müssen mit den zuvor serverseitig gespeicherten Daten übereinstimmen.

- Zwischen Generierung des Spiels und Empfang der gesammelten Daten dürfen nur 15 Sekunden vergangen sein. Diese Zeitspanne ergibt sich aus der maximalen Spieldauer von zehn Sekunden und der Laufzeit, welche für die Seitenaufrufe im mobilen Internet eingeplant werden muss. In wenigen Fällen reicht diese Zeit im mobilen Internet leider nicht aus. Während der Entwicklung wurden hier bis zu 56 Sekunden gemessen. Das Erhöhen der Zeit auf diesen Wert erscheint jedoch zu unsicher.

- Das Spiel muss mindestens zwei Sekunden gespielt worden sein. Dies wird anhand der letzten Bildnummer, die ebenfalls zum Server übertragen wird, und der vergangenen Zeit vom Generieren des Spiels bis zum Empfangen der Daten geprüft. Diese Zeitspanne wurde während der Entwicklung des HIPs festgelegt, da es einem menschlichen Benutzer nicht gelingt, das Spiel in einer kürzeren Zeit zu lösen. Durch diese Maßnahme wird ein Angreifer zusätzlich ausgebremst.

- Es müssen mindestens 7 Sensorwerte gesammelt worden sein. Damit soll es Angreifern erschwert werden, die Sensordaten zu fälschen. So genügt es nicht eine Auslenkung des Sensors mit beispielsweise 10 Grad nach oben, 10 Grad nach links und 10 Grad nach unten zu simulieren, da dies zu maximal vier Werten führt (da der Wert der jeweils anderen Achse 0 beträgt).

- Die Sensordaten müssen eine Varianz größer als fünf aufweisen. Obiges Beispiel führt zu einer Varianz von 3 (0, 10, -10). Auch dies soll den Aufwand für einen Angreifer weiter erhöhen, da mehrere verschiedene Werte verwendet werden müssen.

Die erste Version des HIPs beurteilte den Spieler anhand dieser Kriterien. Da jedoch mit dem ersten Angriff (siehe Abschnitt 5.4.1) die Unsicherheit dieses HIPs belegt werden konnte, wurde folgende Überprüfung implementiert. Erfüllen die empfangenen Daten alle zuvor genannten Kriterien, führt der Server anhand der Startkoordinaten und der gesammelten Sensordaten eine Simulation durch. Befindet sich die Kugel am Ende der Simulation im Loch, so wurde das Spiel erfolgreich gelöst und der Benutzer erhält eine dementsprechende Rückmeldung. Dadurch wird sichergestellt, dass es sich nicht um beliebige Zufallswerte, sondern um die gesammelten und zu diesem Spiel gehörenden Sensordaten handelt. Im Falle eines positiven Spielverlaufs wird die aktuelle Session zerstört, um gegen eventuelle Angriffe (vgl. Abschnitt 3.1) geschützt zu sein. Kann der Benutzer hingegen nicht eindeutig als Mensch erkannt werden, wird die Session nicht zerstört. So ist es beispielsweise möglich, die Fehlversuche eines Benutzers zu zählen und diesen gegebenenfalls von weiteren Versuchen auszuschließen.

5.3.2 Entwicklungsschritte

Die Entwicklung dieses HIP fand in mehreren Schritten statt. So war die erste Version des HIPs noch gegen einen sehr einfachen Angriff ungeschützt, wie in Abschnitt 5.4.1 gezeigt wird. Insgesamt wurden während der Entwicklung drei Smartphones eingesetzt. Dabei handelte es sich um die Samsung Galaxy Modelle S3 und S4 und ein Nokia N8-00. Durch den Einsatz der unterschiedlichen Geräte konnte die verfügbare Grafikleistung getestet werden. Während das Spiel auf einem Galaxy S4 sehr flüssig dargestellt wurde, bewegte sich die Kugel auf einem Galaxy S3 nur sehr langsam und reagierte zudem sehr träge auf eine Lageänderung des Geräts. Damals wurde eine sehr große Spielfläche von 600 x 600 Pixeln und Bilder für die Anzeige der Kugel und des Lochs verwendet. Durch eine Reduzierung der Spielfläche auf 200 x 200 Pixel konnte dieses Problem jedoch schnell gelöst werden. Da bei dieser geringen Auflösung die Bilder nicht mehr sauber dargestellt werden konnten, werden die Kugel und das Loch seitdem ebenfalls von JavaScript direkt auf die Zeichenfläche gezeichnet. Diese Maßnahme reduziert zusätzlich das Datenvolumen, da keine Bilddaten übertragen werden müssen. Auch war die Wahl des Zeichenintervalls eine Frage der Performance der Geräte. Die große Spielfläche konnte nur alle 100 ms neu gezeichnet werden, da die Kugel auf dem Galaxy S3 sonst einen Schweif hinter sich her zog. Die kleine Spielfläche kann jedoch alle 40 ms neu gezeichnet werden, was 25 Bildern pro Sekunde entspricht und somit eine flüssige Bewegung darstellen kann.

Während die Smartphone-Modelle Galaxy S3 und S4 sehr detaillierte Sensordaten mit mehreren Nachkommastellen lieferten, war dies bei dem zum Einsatz gekommenen Nokia N8-00 leider nicht

der Fall. Dort wurde die Neigung lediglich in 15 Grad Schritten an das Spiel weitergegeben. Aufgrund der Tatsache, dass für dieses Modell nur wenige Browser zur Verfügung standen, konnte hierfür keine Lösung gefunden werden. Da der Nokia Browser (Version 8.3.1.4) die Sensordaten überhaupt nicht zur Verfügung stellte, erfolgten die Tests stattdessen mit dem Browser Opera Mobile (Version 12.00.2256), der die Daten in dieser reduzierten Form weitergab. Bei der Galaxy Serie war die Wahl des Browsers unbedeutend, da hier alle getesteten Browser, wie der Android Browser, Google Chrome und Firefox Mobile tadellos funktionierten. Es standen zwei weitere Geräte zur Verfügung, die jedoch nicht verwendet werden konnten. Dabei handelte es sich um ein Windows Phone 7 und 8. Das Problem scheint hierbei jedoch nur der verwendete Browser zu sein, da der Internet Explorer die Sensordaten nicht an das Spiel weitergibt. Leider ist das Angebot von Browsern für dieses Betriebssystem sehr begrenzt, sodass keine anderen kompatiblen Browser für diese Geräte gefunden werden konnten. Weitere inkompatible Geräte werden in Abschnitt 5.5 aufgeführt.

Die Eigenschaften der Spielelemente wurden während der Entwicklung definiert und in einem begrenzten Personenkreis evaluiert. In diesem Kontext wurde festgelegt, dass die Kugel von Rand abprallen sollte, da dies realistischer erscheint. Auch die Empfindlichkeit der Kugel und die Größe der Elemente wurden in diesem Schritt bestimmt. Bereits an dieser Stelle zeigte sich, dass die Empfindlichkeit von Mensch zu Mensch unterschiedlich eingeschätzt wurde, was auch im Ergebnis der Umfrage, in Abschnitt 5.5.2, thematisiert wird.

5.4 Sicherheit des umgesetzten HIPs

Ein wesentlicher Nachteil dieses Ansatzes ist es, dass das Spiel als JavaScript-Code an das Gerät des Benutzers ausgeliefert wird. Das bedeutet, die Startbedingungen sowie das Verhalten des Spiels ist für einen Angreifer einsehbar. Auch kann genau nachvollzogen werden, welche Daten während des Spiels gesammelt und anschließend übertragen werden. Einzig der serverseitige Teil der Erzeugung und Validation der Daten bleibt dem Angreifer verborgen. Deshalb beruht die Sicherheit dieses HIPs darauf, es einem Angreifer so schwer wie möglich zu machen die geforderten Daten bereitzustellen. Werden für einen Angriff die Sensordaten generiert, muss der Angreifer die zuvor genannten Anforderungen erfüllen. Erschwerend kommt hinzu, dass dieses HIP nur auf einem Gerät mit Lagesensor angezeigt wird. Es ist also auf einem Desktop PC nicht verfügbar. Dies könnte zwar von einem Angreifer ausgehebelt werden, ist jedoch auch mit Mehraufwand verbunden. Möchte ein Angreifer die Sensordaten nicht fälschen, so könnte eine Vorrichtung eingesetzt werden, die das Gerät in die entsprechende Richtung neigt. Dieser Ansatz erscheint jedoch zu aufwändig und wird deshalb innerhalb dieser Studie nicht weiter verfolgt.

Die durchgeführten Angriffe basieren auf dem Prinzip, die Sensordaten zu generieren. Das bedeutet, dass nicht die Daten des Sensors verwendet werden, sondern die geforderten Daten durch Code erzeugt werden. Diese Daten werden, wie vom Spiel gefordert, zu einem Array zusammengefügt und an den Server gesendet. Da es sich um Testangriffe handelt, wurde deren Code in den des Spiels integriert. Ein Angreifer hat diese Möglichkeit nicht, könnte dadurch jedoch nicht abgewehrt werden, da es sich um einen geringen Mehraufwand handelt von außen an die Spieldaten heranzukommen. Denn die Startkoordinaten und der genaue Ablauf des Spiels sind im Code als Klartext enthalten. Auf diese Methode wurde jedoch aus Komplexitätsgründen verzichtet. Zudem sind die Angriffe so einfacher nachzuvollziehen.

Um den Weg der Kugel auch während eines Angriffes sehen zu können, weisen die Angriffe, deren Quellcode ebenfalls im Anhang zu finden ist, etwas mehr Code auf, als sie eigentlich benötigen.

Zugunsten der Übersichtlichkeit verwenden diese Angriffe die spieleigenen Funktionen um die Spielelemente anzuzeigen. Ein Angreifer wird diesen Aufwand vermutlich nicht auf sich nehmen und nur den Algorithmus verwenden, der die entsprechenden Daten generiert.

5.4.1 Angriff 1: Zufällige Daten

Wie bereits in Abschnitt 5.3.2 beschrieben, handelt es sich hierbei um einen sehr einfachen Angriff auf die erste Version dieses HIPs. Da zu diesem Zeitpunkt die gesammelten Daten nicht auf ihre Korrektheit sondern nur auf Quantität und Integrität überprüft wurden, konnte das HIP einfach gebrochen werden. Dieser Angriff ist in Anhang 7.6 ab Zeile 1 zu finden und verwendete zufällige Werte in dem Bereich von -5 bis +5. Diese Werte wurden zusätzlich zu den Startparametern des HIPs gesammelt und nach Erreichen einer vordefinierten Anzahl an den Server gesendet. Da diese Daten in geforderter Quantität vorhanden waren und zudem alle anderen Anforderungen wie Startkoordinaten, Varianz und Informationen über benötigte Frames sowie Zeit aufwiesen, bewertete der Server das jeweilige Spiel als gelöst.

Abbildung 5.6(a) zeigt anhand eines Pfads exemplarisch, wie die generierten Daten die Kugel beeinflussen. Dabei wird sichtbar, dass die Daten einer genauen Überprüfung nicht standhalten könnten, da sich die Kugel von ihrer Startposition ① zu ihrer Endposition ② sogar noch weiter von dem Loch entfernte. Nach diesem Angriff wurde das HIP um die serverseitige Überprüfung der Sensordaten mittels einer Simulation erweitert. Dieser Angriff führt seitdem nicht mehr zum Erfolg.

5.4.2 Angriff 2: Definierte Werte

Nach der Implementation der serverseitigen Überprüfung der gesammelten Sensordaten wurde dieser modifizierte Angriff getestet. Damit sollte überprüft werden, ob die nun geforderten Daten korrekt ermittelt und generiert werden könnten, um das HIP auf diese Weise zu brechen. Der Code für diesen Angriff, der in zwei Phasen unterteilt wurde, ist in Anhang 7.6 ab Zeile 28 zu finden.

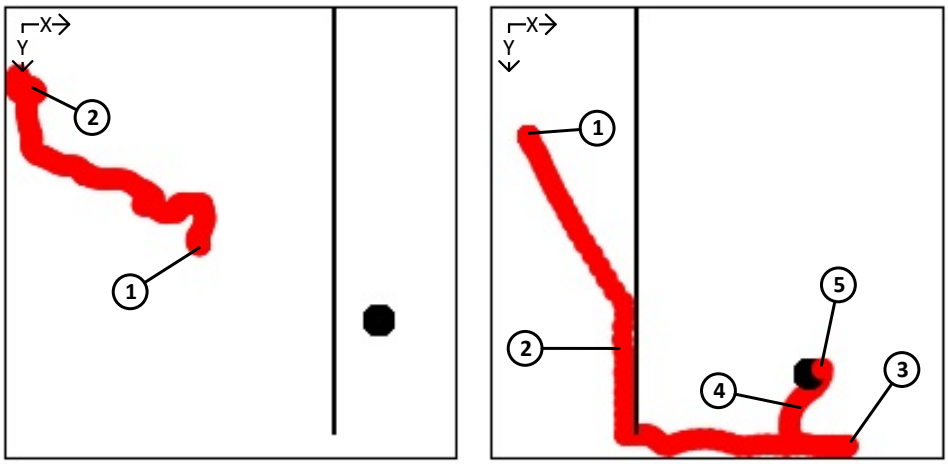
In der ersten Phase werden die Daten gesammelt, die notwendig sind damit sich die Kugel auf den Spalt in der Barriere zubewegt. Dazu wird die Kugel in Richtung der X- und Y-Achsen auf den Spalt zugerollt. Befindet sich das Loch links von der Kugel, so müssen positive Werte für die Bewegung in X-Richtung erzeugt werden, rechts davon negative. Um die Kugel nach unten rollen zu lassen, müssen positive Werte erzeugt werden. Bei diesen Werten handelt es sich um zufällig generierte Werte im Bereich von null bis zehn. Die Richtung der Kugel wird dementsprechend über das Vorzeichen gesteuert. Durch die Zufallszahlen besitzt dieser Angriff keine Kontrolle über die Geschwindigkeit der Kugel, es wird ausschließlich deren Richtung gesteuert.

Je nach Startposition der Kugel trifft sie bereits vor Erreichen des Spalts auf die Barriere oder den Rand des Spielfelds. Da die Kugel abhängig von ihrer Geschwindigkeit von dort wieder zurückprallt, wird sie immer wieder in diese Richtung beschleunigt. Dabei rollt sie an dem Hindernis entlang, bis sie durch den Spalt gelangt. Nach durchqueren des Spalts, wird die Y-Position weiterhin gehalten, während sich die Kugel in X-Richtung immer weiter auf die X-Position des Lochs zubewegt. Stimmen die X-Positionen der Kugel und des Lochs überein, wird die zweite Phase gestartet.

In der zweiten Phase wird die Kugel durch den ständigen Abgleich ihrer X-Position auf Höhe des Lochs gehalten. Auf diese Weise muss die Kugel vor dem Ende der ersten Phase nicht abgebremst werden,

sondern sie wird dementsprechend in die andere Richtung beschleunigt, sollte sie sich wieder von dem Loch entfernen. Gleichzeitig erfolgt das Rollen der Kugel in Y-Richtung auf das Loch zu. Da auch in dieser Richtung ein eventuelles Vorbeirollen der Kugel durch die entgegengesetzte Beschleunigung ausgeglichen wird, gelangt die Kugel in relativ kurzer Zeit in das Loch. Diese Zeit ist in wenigen Fällen durch die exakte Steuerung so kurz, dass dieser Angriff aufgrund dessen fehlschlägt. Durch verkleinern der generierten Werte kann die Geschwindigkeit der Kugel verlangsamt und dieses Problem somit gelöst werden.

Ein exemplarischer Pfad, den die Kugel bei diesem Angriff nehmen kann, ist in Abbildung 5.6(b) gezeigt. Hier lassen sich zudem die einzelnen Phasen gut erkennen: Von der Startposition ① wird die Kugel in Richtung Loch und Spalt gerollt. Dass die Kugel dabei gegen die Barriere stoßen kann, beeinträchtigt diesen Vorgang nicht, sie rollt stattdessen an der Barriere entlang bis der Spalt erreicht ist ②. Bewegt sich die Kugel mit einer sehr hohen Geschwindigkeit, kann es passieren, dass sie über die X-Position des Lochs hinausschießt ③. In diesem Fall wird die zweite Phase nicht direkt gestartet, sondern die Kugel zuerst abgebremst und in die entgegengesetzte Richtung gerollt. Stimmt nun die X-Position der Kugel mit der des Lochs überein, wird die zweite Phase gestartet und die Kugel auch in Y-Richtung auf das Loch zubewegt ④. Dabei wird die X-Position der Kugel ständig weiter korrigiert. Je nach Geschwindigkeit der Kugel kann es auch in dieser Phase passieren, dass die Kugel nicht direkt in das Loch rollt. Dies wird jedoch wiederum durch entgegengesetztes Beschleunigen der Kugel erreicht ⑤.



(a) Ein exemplarischer Pfad, den die zufällig generierten Daten aus Angriff 1 erzeugen. ① Startposition; ② Position der Kugel bei Ablauf der Spielzeit.

(b) Ein exemplarischer Pfad, der von Angriff 2 generiert wurde. ① Startposition; ② erste Phase; ③ Kugel schießt über Ziel hinaus und wird abgebremst; ④ zweite Phase; ⑤ Kugel gelangt in das Loch.

Abbildung 5.6: Diese Pfade visualisieren die Daten, welche von den verschiedenen Angriffen generiert werden.

Das Phänomen, dass die Kugel über das Ziel hinausschießt, wird teilweise von den zufällig generierten Werten verursacht. Denn wenn einige Male hintereinander sehr hohe Zufallszahlen generiert werden, nimmt die Geschwindigkeit der Kugel stark zu. Geschieht dies kurz vor Erreichen der gewünschten Position, bleibt keine Zeit mehr die Kugel zu verlangsamen oder gar zu stoppen. Dieses Problem könnte behoben werden, indem die Entfernung zum Ziel in die Generierung der Zufallszahlen einfließt. Da der Angriff die Kugel innerhalb weniger Sekunden einlocht, muss er nicht zwingend optimiert werden. Darüber hinaus kann das Vorbeirollen der Kugel auch als menschlich gewertet werden.

5.4.3 Zusammenfassung

Durch die beiden durchgeführten Angriffe konnte anschaulich gezeigt werden, mit welchem geringen Aufwand die korrekten Sensordaten generiert werden können. Hierbei handelt es sich jedoch um ein generelles Problem, das immer dann auftritt, wenn Code im Klartext vorliegt. Denn dann ist für einen Angreifer sofort sichtbar, welche Algorithmen verwendet und welche Daten zwischen Client und Server ausgetauscht werden. Dies gilt vor allem für Webanwendungen. Aus diesem Grund konnte dieses HIP nicht alle der in Abschnitt 5.1 aufgestellten Anforderungen erfüllen. So ist es möglich, das HIP in weit mehr als 0,01 Prozent der Versuche erfolgreich zu simulieren. Um das entwickelte HIP abzusichern bestehen verschiedene Möglichkeiten, die im nächsten Abschnitt diskutiert werden.

5.4.4 Möglichkeiten zur Verbesserung der Sicherheit

Dieser Abschnitt beschreibt mögliche Verbesserungen zur weiteren Absicherung des HIPs gegen einen automatisierten Angriff. Bei den meisten dieser Erweiterungen handelt es sich lediglich um Hürden, die einem Angreifer die Analyse des Spiels und der benötigten Daten erschweren sollen. Die letzte hier vorgestelle Verbesserungsmöglichkeit beruht auf der Erweiterung von Webbrowsern um eine Public-Key-Signatur. Damit kann die Authentizität des Kommunikationspartners sichergestellt werden, was nicht nur bei diesem HIP sondern bei Webanwendungen im Allgemeinen einen deutlichen Sicherheitsgewinn bringt.

Mehrere Löcher

Anstelle von einem Loch könnten mehrere Löcher verwendet werden, eingelocht werden darf jedoch immer nur in ein bestimmtes. Die Unterscheidung könnte anhand der Farbe der Löcher geschehen. Um diese vor einem Angreifer geheim zu halten, könnten Bilder verwendet werden, die von einem PHP-Skript generiert werden. Eine anschließende Überprüfung ist möglich, da der Server die Zuordnung der Farben speichert. Der Angreifer kann die Farbe nicht anhand des Bildnamens herausfinden, sondern muss jedes Bild genauer untersuchen.

Da selbst bei dem Einsatz von vier Löchern immer noch eine Chance von 25 Prozent besteht, durch Raten das korrekte Loch zu treffen, kann diese Erweiterung keinen ausreichenden Schutz bieten. Sie kann nur verwendet werden um den Aufwand für einen Angreifer zu erhöhen, sodass dieser möglicherweise das Interesse verliert. Darüber hinaus ist dadurch eine größere Geschicklichkeit gefordert, da die Kugel nicht in die übrigen Löcher rollen darf. Durch die Bildübertragung würde zudem das benötigte Datenvolumen ansteigen.

Verschleiern der Lochposition

Eine andere Möglichkeit besteht darin, die genaue Position des Lochs zu verschleiern. Dies kann ebenfalls durch eine serverseitige Erstellung eines Bildes erfolgen. Jedoch muss in diesem Fall das Bild wesentlich größer als das darin gezeigte Loch sein. Da die Position des Lochs zufällig gewählt wird, kann ein Angreifer die Kugel nicht ohne tiefergehende Analyse einlochen. Allerdings kann so auch das Spiel selbst nicht erkennen, ob die Kugel in das Loch gelangt. Um dies herauszufinden, muss das Spiel die Position der Kugel an den Server senden, sobald diese sich innerhalb des generierten

Bildes befindet. Stimmt die Position der Kugel mit der des Lochs überein, so sendet der Server einen entsprechenden Befehl an das Spiel, das sich daraufhin beendet.

Hierbei entsteht jedoch ein Problem für den Benutzer: Da die Koordinaten des Lochs dem Spiel unbekannt sind und nur im Server vorliegen, kann es dem Benutzer kein direktes Feedback geben. Durch die hohen Latenzen, die vor allem im mobilen Internet auftreten können, würde die Gebrauchstauglichkeit leiden. Denn für den Spieler sieht es im ersten Moment so aus, als würde die Kugel über das Loch gleiten. Darüber hinaus bleibt es für einen Angreifer weiterhin möglich die Position des Lochs aus dem Bild auszulesen, es bedarf nur eines größeren Aufwands.

Cloud gaming

Unter *cloud gaming* versteht man das Verfahren, bei dem die Spiellogik auf dem Server verbleibt und der Client nur die Befehle des Spielers entgegennimmt und an den Server sendet. Der Server errechnet daraus beispielsweise die neuen Positionen der Gegenstände und sendet diese an den Client, welcher diese Daten in Bilder umrechnet und anzeigt. Dieses Vorgehen hat den Vorteil, dass der Quelltext des Spiels immer auf dem Server verbleibt und dort nicht eingesehen werden kann. So ist es beispielsweise bei Online Games nicht möglich einen Spielstand zu fälschen, da der Client lediglich ein erweitertes Ein- und Ausgabegerät darstellt. Dieses Verfahren würde sich auch bei diesem HIP anbieten, da ein Angreifer hierbei nicht auf die Logik des Spiels zugreifen kann. Jedoch ist *cloud gaming* sehr anfällig gegenüber Latenzen und deshalb nur bei rundenbasierten oder Strategiespielen anwendbar [AFG10]. Für ein solches Spiel wie es bei diesem HIP entwickelt wurde, bei welchem unter Umständen sehr schnell reagiert werden muss, sind diese Verzögerungen zu groß.

Parameter verschleiern

Derzeit ist es für einen Angreifer sehr leicht die Parameter des Spieles zu erfahren, da sich diese im JavaScript-Code der HIP-Internetseite befinden. Die Parameter könnten verschleiert werden, indem diese dem Spiel nicht vom ersten Moment an bekannt sind, sondern separat angefragt werden müssen. Diese Anfrage darf jedoch nur einmal gestellt werden, da der Server das Spiel im Falle einer zweiten Anfrage als ungültig markierten würde. Ein Angreifer könnte jedoch auch diese Anfrage automatisieren und so an die erforderlichen Daten gelangen. Mit diesem Vorgehen kann demnach auch nur der Aufwand für einen Angriff erhöht werden.

Erweiterte Sensordaten

Um die zu übertragende Datenmenge klein zu halten, werden nur die für das Spiel verwendeten Sensordaten an den Server gesendet. Jedoch fallen zwischen den Zeichenintervallen des Spiels ebenfalls Daten an. Würden diese Daten ebenfalls zur Auswertung genutzt, ist es unter Umständen möglich, genauere Aussagen zu treffen. Jedoch können auch diese Daten automatisiert generiert und gefälscht werden. Auch der Jitter[3] oder Zeitstempel der Sensoren könnte gefälscht werden.

[3] Jitter bezeichnet in diesem Zusammenhang die Schwankung der Sensordaten um einen bestimmten Wert. Diese Schwankung wird durch die Ungenauigkeit der Sensoren sowie durch das Zittern der Hände verursacht.

Verwendung von künstlicher Intelligenz

Der serverseitige Einsatz von künstlicher Intelligenz könnte möglicherweise dazu verwendet werden, eine Aussage über das Wesen des Spielers zu treffen. Dazu könnten die gesammelten Sensordaten analysiert werden. Derzeit werden jedoch nur die nötigsten Daten an den Server übertragen um Datenvolumen zu sparen. Ob mit dieser begrenzten Anzahl von Daten überhaupt eine Klassifizierung mit angemessener Genauigkeit erreicht werden kann, konnte nicht getestet werden, da hierfür eine Umfrage mit weitaus größerem Umfang benötigt wird. Dies besonders aus dem Grund, da diese Spiele nicht nur von einer Person gespielt werden müssen, sondern von möglichst vielen verschiedenen Personen, um die Klassifizierung nicht auf Basis einzelner Personen aufzubauen. Sonst könnten nur Personen unterschieden werden und nicht zwischen Menschen und Maschinen.

Jedoch besteht auch dann das Risiko, dass ein automatisierter Angriff nicht erkannt werden kann, da er sich zu menschlich verhält. Hieraus würde wiederrum ein Wettlauf gegen die Angreifer entstehen, der unter Umständen nicht gewonnen werden kann. Aufgrund dieser Unsicherheit wurde auf die Umfrage nicht dahingehend ausgerichtet.

Erweiterung des Browsers

Die bereits genannten Verbesserungsmöglichkeiten können das Sicherheitsproblem leider nicht nachhaltig lösen. Auch eine Weiterentwicklung derselben könnte möglicherweise den daraus entstehenden Wettlauf gegen potentielle Angreifer nicht gewinnen. Da mit dieser Studie jedoch ein System entwickelt werden sollte, das einerseits den Zielkonflikt zwischen Sicherheit und Gebrauchstauglichkeit entschärft und zugleich ein Wettrüsten verhindert, bedarf es einer anderen Herangehensweise. Eine Möglichkeit hierfür wird in diesem Abschnitt beschrieben. Durch die Erweiterung der Browser um eine Public-Key-Signatur kann die Authentizität des Kommunikationspartners, in diesem Fall die des HIPs, sichergestellt werden. Einem Angreifer wird es so unmöglich gemacht, den Code des HIPs zu ändern.

Alle modernen Browser weisen die gleiche Architektur auf, wenn auch mit geringen Abweichungen [GG06, Vrb13]. Eine abstrakte Darstellung dieser Architektur ist in Abbildung 5.7 gezeigt. Diese Architektur wurde von Grusskurth und Godfrey [GG06] entwickelt und gegen gängige Browser verifiziert. Diese Abbildung zeigt unter anderem die beiden Instanzen, die für diese Erweiterung wichtig sind. Dabei handelt es sich um das *Networking* Modul, welches die Kommunikation mit dem Server übernimmt, und den *JavaScript Interpreter*, der den JavaScript-Code interpretiert und ausführt.

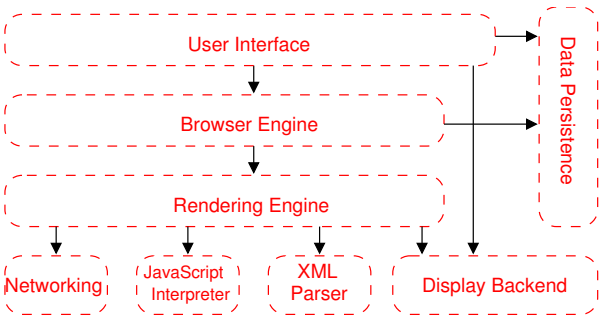


Abbildung 5.7: Eine abstrahierte Darstellung der Architektur moderner Browser (aus [GG06]).

Gelingt es dem Spieler die Kugel in das Loch zu rollen, wird ein *XMLHttpRequest* erstellt und damit die gesammelten Daten an den Server gesendet. Diese Anfrage wird im JavaScript erstellt und an die API

des Browsers gerichtet, welcher daraufhin das *Networking* Modul anweist, diese Anfrage an den Server zu senden. An dieser Stelle muss sichergestellt werden, dass es sich bei diesen Daten um Messwerte des Gyroskops handelt. Um dies zu erreichen, muss der Beweis erbracht werden, dass die Daten durch das Spiel gesammelt wurden. Darüber hinaus muss bewiesen werden, dass das Spiel nicht manipuliert wurde. Dieser Beweis muss jedoch von einer höheren Instanz als dem Spiel selbst geliefert werden, da jegliche Implementierung innerhalb des Spiels von einem Angreifer gelesen und nachgebaut werden kann. Als höhere Instanz ist der Browser am naheliegendsten. Dessen API könnte um eine Funktion erweitert werden, die ebenfalls vom JavaScript des Spiels aufgerufen werden kann. Diese Funktion nimmt, genau wie die Funktion, die vom *XMLHttpRequest* aufgerufen wird, die zu sendenden Daten entgegen. Zusätzlich bildet diese neue Funktion jedoch einen Hash-Wert des sendenden JavaScript-Codes. Dadurch kann überprüft werden, ob es sich um ein unmanipuliertes Spiel handelt, da bereits die kleinste Veränderung des Codes einen anderen Hash-Wert ergibt.

Da dieser Hash-Wert ebenfalls von einem Angreifer gefälscht werden könnte, muss der Browser die zu sendenden Sensordaten sowie den ermittelten Hash-Wert digital signieren. Hierfür benötigt der Browser ein Public-Key-Zertifikat. Es ist jedoch nicht notwendig für jeden Browser ein eigenes Zertifikat zu verwenden. Es würde genügen, beispielsweise ein Zertifikat für *trusted browsers* oder *well known browsers* zu erstellen, welches sich die Browserhersteller teilen. Denn es ist für diesen Anwendungsfall nicht von Belang, welcher Browser den Server kontaktiert, sondern nur, dass es sich dabei nicht um einen manipulierten Browser handelt, der beispielsweise den Hash-Wert nicht selbst errechnet. Der Ablauf dieses Verfahrens ist in Abbildung 5.8 gezeigt.

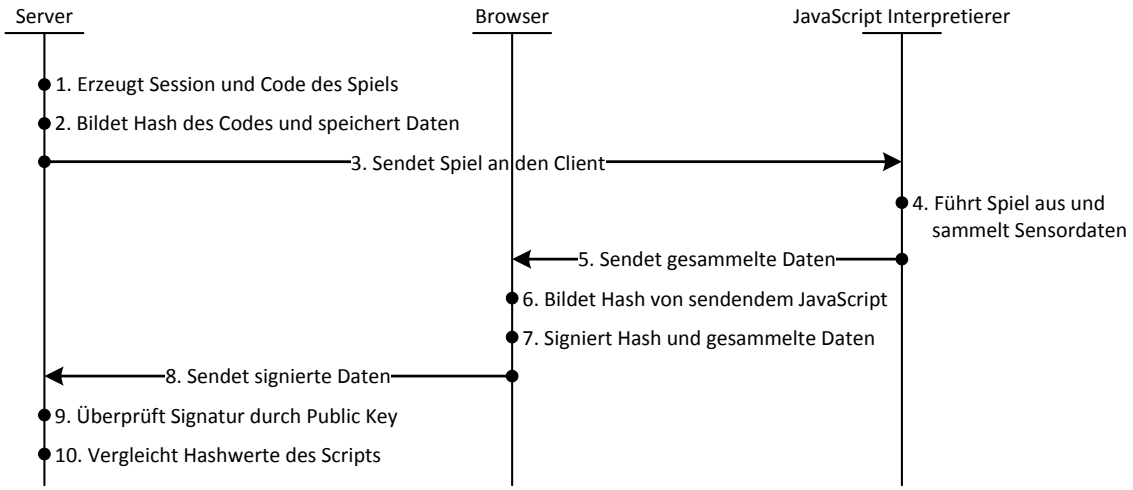


Abbildung 5.8: Ablauf der Kommunikation mit der Hash- und Signatur-Erweiterung.

Das HIP unterstützt bereits diese Erweiterung, hierfür wurde die serverseitige Generierung des Spiels angepasst. In Anhang 7.1, Zeile 47-56 ist zu sehen, wie die zufälligen Startparameter in den Code des Spiels eingefügt werden. Dieser wird aus Gründen der Übersichtlichkeit aus der Datei *HIP.js* (siehe Anhang 7.2) gelesen und im Anschluss an die Parameter eingefügt. Danach wird über den gesamten Code ein Hash-Wert gebildet. Da in diesem Code auch die Startparameter enthalten sind, ändert sich auch der Hash-Wert mit jedem Spiel. Dieser Hash-Wert wird serverseitig in der Session gespeichert, und kann später mit dem vom Browser errechneten Hash-Wert verglichen werden.

Im JavaScript-Code des HIPs wurde der Aufruf der neuen Browserfunktion bereits beispielhaft implementiert (siehe Anhang 7.2, Zeile 116ff). Hierzu wurde eine neue Klasse namens *SecureHttpRequest*

von *XMLHttpRequest* abgeleitet, die diesen jedoch um die bereits genannten sicherheitsrelevanten Ergänzungen erweitert. Da es nicht möglich ist, innerhalb des JavaScript-Codes den Hash-Wert desselben zu berechnen, wurde diese Funktion nicht implementiert, sondern ist nur durch Pseudocode innerhalb eines Kommentars in der Klasse *SecureHttpRequest* dargestellt (siehe Anhang 7.3).

Die Überprüfung des Hash-Werts und der Signatur des Browsers ist in Anhang 7.4, Zeile 21-30 gezeigt. Dazu wird die empfangene Nachricht mit dem öffentlichen Schlüssel des Browsers entschlüsselt und anschließend der Hash-Wert geprüft. Gelingt das Entschlüsseln und stimmt der Hash-Wert mit dem zuvor gespeicherten überein, so können die Daten den bereits bekannten Überprüfungsmaßnahmen (vgl. Abschnitt 5.3.1) unterzogen werden. Denn somit ist sichergestellt, dass es sich um Sensordaten handelt, die auf die geforderte Weise gesammelt wurden.

Diese Erweiterung kann nicht nur auf dieses HIP angewendet werden, sondern generell immer dann, wenn es möglich sein soll die Quelle von Daten überprüfen zu können. Dies betrifft einerseits Sensordaten, andererseits kann auf diese Weise aber auch gewährleistet werden, dass beispielsweise ein Kontaktformular nur mittels eines Browsers ausgefüllt werden kann. Diese Fähigkeit würde die Möglichkeiten von Angreifern einschränken, da diese nicht mehr mittels eines Skriptes Spam-Nachrichten versenden könnten, sondern dies mittels eines Browsers tun müssten. Dadurch könnte in vielen Bereichen auf den Einsatz von HIPs komplett verzichtet werden.

5.5 Gebrauchstauglichkeit des umgesetzten HIPs

Dieser Abschnitt beschreibt das angewendete Vorgehen um die Gebrauchstauglichkeit des HIPs zu überprüfen. Bereits während der Entwicklung des HIPs wurde auf eine gute Gebrauchstauglichkeit geachtet. Aus diesem Grund wurden die Entwicklungsstadien von mehreren Personen auf unterschiedlichen Geräten getestet. Insgesamt wurden drei Smartphones (Samsung Galaxy S3, S4, Nokia N8-00) eingesetzt, die neben dem Entwickler von drei anderen Personen verwendet wurden. Dadurch konnte, wie bereits in Abschnitt 5.3.2 beschrieben, die Größe der Spielfläche und Elemente, das Zeichenintervall und die Empfindlichkeit der Kugel optimiert werden.

Auch wurde das HIP auf die in Abschnitt 5.1 aufgestellten Anforderungen hin überprüft: Durch die Verwendung der Lagesensoren wird keine Bildschirmtastatur verwendet und der Benutzer kann das HIP zu jeder Zeit vollständig betrachten. Das HIP wurde in Form eines Spiels entwickelt und wird im Browser angezeigt, darüber hinaus ist keine Installation einer zusätzlichen Software notwendig und es wird kein Text verwendet, wodurch keine Sprachbarriere entsteht. Auch werden keine Kulturkreise ausgeschlossen, da das Spiel einzig auf den Gesetzen der Physik beruht, die weltweit gültig sind. Durch eine Analyse des Datenverkehrs zwischen dem Client und dem Server (siehe Abbildung 5.5) konnte ermittelt werden, dass das ausgetauschte Datenvolumen mit weniger als 10 kB sehr gering ausfällt und das HIP ohne Bedenken für die mobile Nutzung verwendet werden kann. Diese geringe Datenmenge konnte zum Teil auch durch den ausschließlichen Einsatz von JavaScript erreicht werden. So muss lediglich der Code des Spiels übertragen werden, da der JavaScript-Interpretierer bereits auf den mobilen Endgeräten vorhanden ist. Würde ein zusätzliches Framework (bspw. PhoneGap) verwendet, müsste dieses zusätzlich zum Spiel heruntergeladen werden. Aufgrund der Bauweise der Smartphones konnte die Erhöhung der Stromaufnahme durch das Spiel nicht gemessen werden. Während der Entwicklungsphase war das Spiel jedoch in Einzelfällen sehr lange geöffnet. Jedoch konnte keine signifikante Verkürzung der Akkulaufzeit festgestellt werden. Aus diesem Grund wird davon ausgegangen, dass sich die Leistungsaufnahme durch das Spiel in einem akzeptablen Bereich befindet.

Neben der Einhaltung dieser Anforderungen wurde die Gebrauchstauglichkeit des HIPs durch eine Onlineumfrage überprüft. Der Ablauf und Umfang dieser Umfrage wird im nächsten Abschnitt beschrieben. Darauf folgen das Ergebnis der Umfrage und eine Zusammenfassung von möglichen Verbesserungen hinsichtlich der Gebrauchstauglichkeit des HIPs.

5.5.1 Umfrage zur Evaluation der Gebrauchstauglichkeit

Die Umfrage wurde online mittels einer Webseite durchgeführt. Dabei wurde auf der Startseite beschrieben, was das Ziel dieses Projektes ist und wie die Umfrage genau ablaufen wird: Nach dem Start der Umfrage kann der Teilnehmer das Spiel sooft spielen wie er möchte. Nach jedem Einlochen der Kugel kann er wählen, ob er nochmals spielen oder die abschließenden Fragen beantworten möchte. Dadurch sollte getestet werden, wie unterhaltsam das Spiel ist und wie viel Spaß es dem Spieler bereitet. Es wurde davon ausgegangen, dass es einen Zusammenhang zwischen der Anzahl der gespielten Spiele und diesen Faktoren gibt.

Mit den abschließenden Fragen sollte herausgefunden werden, wie geübt der Teilnehmer im Gebrauch mit dem verwendeten Gerät ist und ob er es häufig für das Surfen im Internet verwendet. Darüber hinaus sollten einige Eigenschaften des Spiels abgefragt werden und zudem, wie der jeweilige Teilnehmer das Spiel empfand. Die gestellten Fragen und Antwortmöglichkeiten lauteten wie folgt. Den Antworten wurden Zahlenwerte zugewiesen, um bei der Auswertung den Durchschnitt bestimmen zu können. Darüber hinaus konnten die Teilnehmer zusätzliche Bemerkungen als Freitext angeben.

- Mit welchem Gerät/Modell haben Sie das Spiel getestet?
  Antwortmöglichkeit: Freitext.
- Handelt es sich dabei um Ihr eigenes Gerät?
  Antwortmöglichkeiten: Ja; Nein.
- Falls ja, wie viele Stunden pro Woche nutzen Sie das Internet von diesem Gerät aus?
  Antwortmöglichkeiten: weniger als 1 Stunde; 1-2 Stunden; 2-5 Stunden; 5-10 Stunden; mehr als 10 Stunden. Wertigkeit 1 - 5.
- Drehte sich während des Spiels die Anzeige?
  Antwortmöglichkeiten: Ja; Nein.
- Falls ja, funktionierte dabei die Steuerung weiterhin korrekt?
  Antwortmöglichkeiten: Ja; Nein.
- Wie flüssig bewegte sich die Kugel?
  Antwortmöglichkeiten: ruckelte sehr; ruckelte; mittel; flüssig; sehr flüssig. Wertigkeit 1 - 5.
- Wie empfindlich reagierte die Kugel auf Ihre Bewegung des Geräts?
  Antwortmöglichkeiten: zu träge; träge; ok; empfindlich; zu empfindlich. Wertigkeit 1 - 5.
- War die vorgegebene Zeit ausreichend?
  Antwortmöglichkeiten: Ja; Nein.
- Wie sehr hat Ihnen das Spiel Spaß gemacht?
  Antwortmöglichkeiten: überhaupt nicht; wenig; mittel; viel; sehr viel. Wertigkeit 1 - 5.

Während der Umfrage wurde ebenfalls gespeichert, wie das jeweilige Spiel beendet wurde. Hierfür gibt es drei Möglichkeiten: Der Teilnehmer schafft es nicht rechtzeitig die Kugel einzulochen, oder er kann nach dem rechtzeitigen Einlochen der Kugel als Mensch erkannt werden. Die dritte Möglichkeit besteht darin, dass der Algorithmus den Teilnehmer nach dem Einlochen der Kugel nicht sicher als Mensch erkennt. Diese Informationen sollten zeigen, wie häufig und weshalb das Spiel nicht zum gewünschten Erfolg führte. Mit der Umfrage sollte auch getestet werden, ob das Spiel ohne Anleitung zu verstehen ist. Aus diesem Grund war der Link zur Anleitung versteckt und wurde erst angezeigt, wenn der Teilnehmer die Kugel zweimal nicht innerhalb der Zeit einlochen konnte.

Um Informationen über die verwendeten Geräte zu sammeln wurde mittels folgenden PHP-Befehls jene Zeichenkette gespeichert, mit der sich ein Browser gegenüber Internetseiten identifiziert.

```
$_SERVER["HTTP_USER_AGENT"]
```

Diese Zeichenkette enthält sowohl Informationen über Browsernamen und -version als auch über das Betriebssystem. Im Fall des Apple iPhone 5s sind folgende Informationen enthalten:

```
Mozilla/5.0 (iPhone, CPU iPhone OS 7_0_5 like Mac OS X) AppleWebKit/537.51.1 (KHTML,
    like Gecko) Version/7.0 Mobile/11B601 Safari/9537.53
```

Hieraus ist ersichtlich, dass auf dem Gerät das Betriebssystem iOS 7.0.5 installiert ist und der Browser Mobile Safari (Build 9537.53) verwendet wurde, um die Umfrage aufzurufen. Zwar können diese Informationen gefälscht werden, jedoch konnten bei der Umfrage diesbezüglich keine Auffälligkeiten festgestellt werden. Diese Methode wurde mehrheitlich dazu verwendet, Informationen über inkompatible Geräte zu sammeln, wie im folgenden Abschnitt näher erläutert wird.

5.5.2 Ergebnis der Umfrage

Die Umfrage wurde insgesamt 119 Mal gestartet, jedoch schlossen nur 19 Teilnehmer die Umfrage ab und beantworteten die gestellten Fragen annähernd vollständig. Mindestens weitere 24 Teilnehmer konnten nicht an der Umfrage teilnehmen, da ihr Gerät oder der verwendete Browser nicht kompatibel waren, was gleich zu Beginn der Umfrage überprüft wurde. In einem solchen Fall wurde die Umfrage automatisiert beendet und Informationen über das Gerät und die Software gespeichert.

Über diese Zahl hinaus wurden dem Entwickler Fälle bekannt, bei denen die Umfrage nicht abgebrochen wurde, das Spiel aber dennoch nicht gespielt werden konnte oder gar nicht angezeigt wurde. Mögliche Gründe hierfür sind fehlende Sensoren im Gerät sowie ein veraltetes Betriebssystem oder Browser. Ein Fall trat auf, in dem das Spiel zwar angezeigt wurde, sich die Kugel jedoch nicht bewegen ließ. Hierbei unterstützte der Browser (Firefox Mobile) aufgrund seiner Aktualität zwar das geforderte *DeviceOrientationEvent*, jedoch verfügte das Gerät über keinerlei Sensoren, sodass das Event niemals ausgelöst wurde. Dabei handelte es sich um ein Tablet der Marke MSI mit installiertem Windows 8. Bei älteren iPods und iPads wurde das Spiel aufgrund eines *Application Instantiation Error* nicht gestartet. Was diesen Fehler verursacht, konnte nicht festgestellt werden. Darüber hinaus konnte das Spiel auf einem iPhone 3 aufgrund eines veralteten Betriebssystems ebenfalls nicht gespielt werden. Bei diesem Gerät konnte jedoch die Inkompatibilität erkannt werden, was das Anzeigen eines alternativen HIPs möglich machen würde. Auf dem iPad 3 wird das Spiel zwar angezeigt, jedoch ist die Bewegung der beiden Achsen invertiert. Das Spielen ist so war möglich, führt aber zu einer erheblichen Verwirrung des Spielers. Diese Anomalie wurde vermutlich durch eine veraltete Version des Browsers verursacht, da die Ausrichtung der Achsen auch derzeit noch nicht ausreichend standardisiert sind [Moz].

Es liegt demnach auf der Hand, dass für das Spielen des HIPs ein aktueller Browser enorm wichtig ist. Denn selbst wenn Gerät und Betriebssystem die Daten an den Browser liefern, hängt es von diesem ab, sie an das HIP weiterzuleiten. Ein gutes Beispiel hierfür ist das Smartphone Nokia N8-00: Das Gerät sowie das Betriebssystem sind kompatibel, jedoch kann das Spiel nicht mit dem mitgelieferten Nokia Browser, sondern nur mit Opera Mobile gespielt werden. Dieser Browser gibt jedoch nicht alle gemessenen Werte weiter, sondern nur in 15 Grad Schritten. Deshalb ist die Steuerung des Spiels schwieriger als auf den anderen Geräten. Ob sich dies mit der nächsten Version des Opera Mobile Browsers ändert, konnte nicht festgestellt werden. Ein weiteres Problem konnte bei dem Dolphin Browser festgestellt werden, der zwar das Spiel korrekt anzeigte und ausführte, jedoch entsprachen die gesendeten Daten, wie beispielsweise die initialen Koordinaten, nicht den serverseitig gespeicherten. Die Ursache für dieses Verhalten konnte nicht identifiziert werden.

Die Tabellen 5.1 und 5.2 zeigen im Einzelnen welche Browser und Betriebssysteme durch die Umfrage getestet wurden. Ist ein Browser oder Betriebssystem kompatibel, so stellt die angegebene Versionsnummer die älteste getestete Version dar. Es ist möglich, dass bereits noch ältere Versionen kompatibel sind, diese wurden jedoch von keinem Teilnehmer der Umfrage verwendet. Sollte ein Produkt nicht für dieses Spiel geeignet sein, beschreibt die angegebene Version die aktuellste, die getestet wurde. Generell kann gesagt werden, dass dieses HIP auf allen aktuellen Geräten, mit Ausnahme von Windows Phones, voll funktionsfähig ist. So wurden bei iPhones der Version 4 und 5 keine Probleme festgestellt. Auch die Android Geräte wie Samsung Galaxy S2, S3 und S4 funktionierten einwandfrei, sofern eine entsprechende Version des Android Betriebssystems installiert war.

Browser	Kompatibel	Version
Android Browser	Ja	ab 4.0.4
Chrome	Ja	ab 30.0
Dolphin Browser	Eingeschränkt	ab 10.2.6
Firefox Mobile	Ja	ab 4.0
Internet Explorer	Nein	bis 10.0
Mobile Safari	Ja	ab 534.30
Nokia Browser	Nein	bis 8.3.1.4
Opera Mobile	Eingeschränkt	ab 12.00

Tabelle 5.1: Kompatibilität der getesteten Browser.

Betriebssystem	Kompatibel	Version
Android	Ja	ab 4.0.4
iOS	Ja	ab 7.0.4
Symbian	Ja	ab S60 5.3
Windows Phone	Nein	bis Windows Phone 8.0

Tabelle 5.2: Kompatibilität der getesteten Betriebssysteme.

Die Auswertung der Daten, die während der Spiele erfasst wurden, ist in Tabelle 5.3 zu finden. Die Interpretation und weitere Erläuterungen hierzu folgen im weiteren Verlauf. Die Anforderung an ein HIP, dass es vom Menschen in 80 Prozent der Versuche korrekt gelöst werden kann, wurde bei dieser Umfrage übertroffen. So konnten die Teilnehmer der Umfrage 61 von 69 Spielen korrekt lösen, was 88,4 Prozent entspricht. Drei der ungültig abgeschlossenen Spiele wurden auf einem Nokia N8-00 gespielt. Durch die sehr eingeschränkten Sensordaten konnte die Anforderung an die Varianz der Daten nicht erfüllt und somit nicht sichergestellt werden, dass es sich hierbei um einen Menschen handelt. Die übrigen vier ungültigen Spiele wurden auf einem Galaxy S4 mit Dolphin Browser gespielt, welcher die Daten nicht wie gefordert sendete. Diese Ausnahmen zeigen, dass das Spiel mit einem vollständig kompatiblen Browser und Betriebssystem in nahezu 100 Prozent der Versuche gelöst werden kann.

Jeder Teilnehmer spielte durchschnittlich zwischen drei und vier Spiele. Sechs Teilnehmer spielten jeweils nur einmal, diese gaben jedoch den Spaßfaktor mit „mittel“ an, was demnach nicht zwingend die Ursache für die geringe Anzahl sein muss. Die maximale Anzahl von neun Spielen wurde auf einem iPhone gespielt. Das einzige Spiel, das im Rahmen der Umfrage nicht durch das rechtzeitige einlochen der Kugel beendet werden konnte, wurde ebenfalls von diesem Teilnehmer gespielt (siehe Tabelle 5.3).

Mit durchschnittlich 6,5 Sekunden pro Spiel war dieser Teilnehmer etwas langsamer als die mittlere Spieldauer aller Teilnehmer von 5,9 Sekunden, bewertete das Spiel aber dennoch mit dem Spaßfaktor „viel“. Der Fehlversuch und die allgemein längere Spieldauer scheinen demnach das Spielvergnügen nicht beeinflusst zu haben.

Da während der gesamten Umfrage nur ein Fehlversuch durch Zeitüberschreitung auftrat, wurde der Link zur Anleitung nie angezeigt, und die Anleitung demnach niemals aufgerufen. Dies zeigt, dass das Spiel auch ohne eine Anleitung nahezu problemlos gelöst werden kann. Wenige der Teilnehmer gaben jedoch an, dass ihnen zunächst die Aufgabe bzw. das Ziel des Spiels nicht klar war, und sie sich deshalb eine Anleitung gewünscht hätten. Durch das Ausblenden des Links zur Anleitung während der Umfrage, sollte geklärt werden, ob das Spiel auch ohne diese verständlich und lösbar ist. In der normalen Version des HIPs ist dieser Link selbstverständlich immer sichtbar.

Spiele insgesamt	69	
Davon erfolgreich	61	(88,4%)
Davon Zeitüberschreitung	1	(0,01%)
Davon ungültig	7	(0,10%)
Spiele pro Teilnehmer	3,6	$\sigma = 2,65$
Durchschnittliche Dauer	5,9 sek	$\sigma = 1,90$
Aufrufe der Anleitung	0	

Tabelle 5.3: Auswertung der Spieldaten der Umfrage.

Nach jedem Einlochen konnten die Teilnehmer ein neues Spiel starten oder weiter zur Umfrage gelangen. In den folgenden Absätzen werden die Ergebnisse dieser abschließenden Fragen interpretiert, die in Tabelle 5.4 zusammengefasst sind. Alle Teilnehmer verwendeten für die Umfrage ihr eigenes Gerät, wobei sie dieses Gerät im Mittel 2-5 Stunden pro Woche für die Internetnutzung verwendeten. Nur in zwei Fällen (iPhone 5, Xperia mini) drehte sich während des Spielens die Anzeige, die Steuerung funktionierte jedoch in jedem Fall weiterhin korrekt. Ein Teilnehmer gab an, während dem Lösen der Aufgabe die Displaydrehung deaktiviert zu haben. Dies ist zwar auf iPhones und Android Geräten möglich, gilt aber nicht für alle an der Umfrage beteiligten Geräte. Dieses Ergebnis bestätigt dennoch die Annahme, dass das Spiel auch durch so leichte Neigungen gelöst werden kann, dass die Anzeige nicht automatisch gedreht wird.

Mit der Frage „Wie flüssig bewegte sich die Kugel?“ sollte die Performance auf den jeweiligen Geräten geprüft werden. Wie bereits beschrieben, wurde während der Entwicklungsphase mit der Größe der Spielfläche und dem Zeichenintervall experimentiert. Laut der Umfrage bewegte sich die Kugel durchschnittlich „flüssig“, was ein gutes Ergebnis darstellt. Auch die Empfindlichkeit der Kugel war laut der Umfrage „ok“, jedoch gingen hier die Meinungen der Teilnehmer auseinander. Es konnte hierbei eine Tendenz bezüglich der verwendeten Geräte erkannt werden: Während Benutzer von iPhones mit einem Wert von 2,8 in Richtung „träge“ tendierten, gaben die Besitzer von Android Geräten einen Wert von 4,2, also „empfindlich“ an. Ein Benutzer eines iPhones wünschte sich sogar eine Steigerung der Schwierigkeit im Spielverlauf, während ein Benutzer eines Samsung Gerätes angab, es sei leicht die Kugel um die Barriere herum zu führen, jedoch gelinge das Einlochen der Kugel aufgrund der hohen Empfindlichkeit nur sehr schwer. Aus diesem Grund sollte eine geräteabhängige Empfindlichkeit angestrebt werden.

83 Prozent der Teilnehmer gaben an, die Zeit als ausreichend zu empfinden. Die anderen 17 Prozent waren alle Benutzer eines Android Gerätes. Das legt nahe, dass diese Benutzer durch die hohe Empfindlichkeit dieser Geräte mehr unter Zeitdruck geraten sind, als die Benutzer anderer Geräte. Zwar gelang

es allen, das Spiel dennoch innerhalb der vorgegebenen Zeit zu lösen, jedoch sollte das unangenehme Gefühl, unter Zeitdruck zu stehen, vermieden werden. Ob dies durch eine geräteabhängige Anpassung der Empfindlichkeit gelöst werden kann, sollte in einer weiteren Umfrage ermittelt werden. Diese Option sollte jedoch auch im Hinblick auf die Sicherheit näher beleuchtet werden, da hierdurch ein Sicherheitsrisiko entstehen könnte.

Die Frage nach dem Spaßfaktor des Spiels beantworteten die Teilnehmer mit einem Wert von 3,5 als „mittel“ bis „viel“. Bei dieser Frage konnte keine Tendenz von den Benutzern der verschiedenen Geräte, sowie keine weiteren geräte- oder softwarespezifischen Gründe für die jeweiligen Antworten gefunden werden. Die Meinung zu dieser Frage scheint auf persönlichen Präferenzen zu beruhen. Nur zwei Teilnehmer gaben an, das Spiel mache „wenig“ Spaß, keine Antwort lautete „überhaupt nicht“. Aufgrund dieser Antworten scheint das, im Rahmen dieser Arbeit entwickelte, HIP eine angenehme Alternative zu den bisher verwendeten HIPs zu sein.

Frage	Durchschnittliche Antwort
Handelt es sich dabei um Ihr eigenes Gerät?	Ja (100%)
Wie viele Stunden pro Woche nutzen Sie das Internet mit diesem Gerät?	2-5 Stunden (3,16)
Drehte sich während des Spiels die Anzeige?	Nein (98%)
Funktionierte dabei die Steuerung weiterhin korrekt?	Ja (100%)
Wie flüssig bewegte sich die Kugel?	Flüssig (4,2)
Wie empfindlich reagierte die Kugel auf Ihre Bewegung des Geräts?	Ok (3,4)
War die vorgegebene Zeit ausreichend?	Ja (83%)
Wie sehr hat Ihnen das Spiel Spaß gemacht?	Mittel - Viel (3,5)

Tabelle 5.4: Durchschnittliche Antworten auf die gestellten Fragen.

Während der Umfrage wurde ebenso überprüft, ob die Latenzzeiten des mobilen Internets möglicherweise zu hoch sind, um die vorgegebenen 15 Sekunden einhalten zu können. Es trat jedoch in keinem Fall ein diesbezüglicher Fehler auf. Die Art der Internetverbindung der Geräte konnte während der Umfrage nicht geprüft werden, es wird jedoch davon ausgegangen, dass die Teilnehmer das Internet wie gewohnt genutzt haben. Dies lässt den Schluss zu, dass das Zeitlimit von 15 Sekunden auch im realen Einsatz eingehalten werden kann.

5.5.3 Möglichkeiten zur Verbesserung der Gebrauchstauglichkeit

Dieser Abschnitt fasst die Verbesserungen zusammen, die im Hinblick auf die Gebrauchstauglichkeit durch die Umfrage gefunden werden konnten. Durch die Anwendung dieser Verbesserungen können alle Anforderungen, welche in Abschnitt 5.1 an die Gebrauchstauglichkeit gestellt wurden, erfüllt werden.

Die Kompatibilitätsprüfung, die derzeit auf Abfrage des *DeviceOrientationEvent* Events beruht, sollte dringend erweitert werden, um Sonderfälle wie die des iPhone 3 erkennen zu können. Bei diesem Gerät war die Messung der Neigung invertiert. Dies könnte möglicherweise anhand der Version von Betriebssystem oder Browser erkannt und programmatisch behoben werden. Wird das Event niemals ausgelöst, weil möglicherweise keine Lagesensoren verbaut sind, könnte dies ebenfalls erkannt und stattdessen ein alternatives HIP angezeigt werden.

Im Zuge der Geräteerkennung sollte auch die Empfindlichkeit der Kugel entsprechend angepasst werden. Während der Umfrage wurde festgestellt, dass bei Geräten mit Android System die Kugel empfindlicher zu reagieren scheint. Ob damit der gefühlte Zeitdruck, der ebenfalls vor allem bei diesen Geräten entstanden war, reduziert werden kann, sollte durch eine weitere Umfrage ermittelt werden. Gegebenenfalls könnte die Spielzeit geringfügig verlängert werden.

Die Drehung der Anzeige für dieses HIP automatisch zu deaktivieren wäre wünschenswert, ist jedoch derzeit nicht möglich. Falls zukünftige Browser hierfür eine Schnittstelle bieten, sollte dies implementiert werden. Allerdings wurde gezeigt, dass das Spiel auch nach einer Drehung der Anzeige weiterhin spielbar ist. Das automatische Deaktivieren der Drehung würde andererseits die Spielbarkeit sicherstellen und Irritationen des Spielers verhindern.

6 Fazit und Ausblick

Seit der Erfindung von HIPs wurden viele unterschiedliche Techniken entwickelt, von denen sich jedoch nur wenige durchsetzen konnten. So wurden beispielsweise alle bildbasierten HIPs, die auf Fotografien beruhen, inzwischen gebrochen. Am häufigsten werden derzeit textbasierte HIPs eingesetzt. An zweiter Stelle stehen bildbasierte HIPs, die generierte Bilder verwenden. Alle HIP-Techniken - mit Ausnahme der spielbasierten - stellen den Benutzer vor eine Prüfungssituation und hinterlassen dadurch zumeist einen negativen Eindruck. Spielbasierte HIPs versuchen den Benutzer hingegen zu unterhalten, weshalb es um deren Akzeptanz besser bestellt ist.

Durch immer ausgefeiltere Angriffe entstand schon bald nach dem Einsatz der ersten HIPs ein Wettlauf zwischen den Angreifern und den Entwicklern der HIPs. Aus diesem Wettlauf heraus entstand ein Zielkonflikt zwischen der Sicherheit und Gebrauchstauglichkeit, der bis heute besteht. Denn während ein einfach zu lösendes HIP eine sehr gute Gebrauchstauglichkeit aufweisen kann, ist dieses mit sehr großer Wahrscheinlichkeit nicht sicher genug. Wird wiederum versucht, die Aufgabe so schwer wie möglich zu gestalten, damit diese von einem automatisierten Angriff gar nicht oder nur sehr selten korrekt gelöst werden kann, liegt nahe, dass ein Mensch diese Aufgabe wenn überhaupt nur mit großer Anstrengung lösen kann. So wird beispielsweise bei textbasierten HIPs versucht, durch Verkrümmung und Überlappung der Zeichen ein automatisiertes Lösen zu verhindern. Durch diese Maßnahmen wird jedoch auch die Lesbarkeit stark beeinträchtigt.

Spielbasierte HIPs scheinen ein möglicher Ausweg aus diesem Wettlauf zu sein. Da hiermit viel komplexere Aufgaben gestellt werden können, müssen diese nicht künstlich erschwert werden. Mit dieser HIP-Technik kann möglicherweise auch der Zielkonflikt zwischen Sicherheit und Gebrauchstauglichkeit gelöst werden. Die Aufgaben, denen sich der Benutzer zum Lösen eines spielbasierten HIPs stellen muss, wirken einfacher, auch wenn wesentlich mehr Interaktion gefordert ist. Durch diese gesteigerte Interaktion bestehen jedoch auch viel mehr Möglichkeiten, an denen eine Unterscheidung bezüglich Mensch oder Maschine festgemacht werden kann. Diese HIPs vermögen es demnach, mit einer besseren Gebrauchstauglichkeit ein höheres Maß an Sicherheit zu erzielen.

Mit dieser Arbeit sollte gezeigt werden, ob die Sensoren mobiler Endgeräte für ein neuartiges HIP verwendet werden können. Zugleich sollte dieses HIP durch Gamifizierung eine größere Akzeptanz finden. Dazu wurden Angriffe auf verschiedenste HIPs untersucht und aus den so gefundenen Schwachstellen Richtlinien abgeleitet, welche für die Entwicklung eines sicheren HIPs notwendig sind. Darüber hinaus wurden Untersuchungen bezüglich der Gebrauchstauglichkeit der verschiedenen HIP-Techniken analysiert und die Erkenntnisse hieraus ebenfalls zu Richtlinien zusammengefasst. Damit konnte ein Rahmenwerk für die Gestaltung eines sicheren und gebrauchstauglichen HIPs formuliert werden. Anhand dieses Rahmenwerks wurde ein Prototyp eines neuen spielbasierten HIPs implementiert. Mit diesem Prototyp konnte Klarheit über die Sicherheit und Gebrauchstauglichkeit eines HIPs für mobile Endgeräte gewonnen werden.

Durch eine Onlineumfrage konnte gezeigt werden, dass es möglich ist, ein spielbasiertes HIP zu entwickeln, welches in mehr als 80 Prozent der Versuche von verschiedenen Personen in sehr kurzer Zeit gelöst werden kann. Zudem reichte die Leistungsfähigkeit der kompatiblen Geräte für die Ausführung dieses HIPs aus. Dies sollten bereits die zahlreichen Tests sicherstellen, die während der Entwicklungs-

phase durchgeführt wurden. Darüber hinaus konnte bewiesen werden, dass ein solches HIP durch die Sensoren mobiler Endgeräte gesteuert werden kann, wenngleich die Anzahl der Sensoren, die mittels HTML5 verwendbar sind, gering ist. Eine weitere Beschränkung stellt die oftmals veraltete Software der mobilen Endgeräte dar, die diese Funktionen ebenfalls nicht unterstützt. Das positive Ergebnis der Umfrage deutet jedoch darauf hin, dass ein solches HIP wesentlich mehr Spaß bereitet als klassische HIPs und überdies ohne eine Anleitung und sonstige Verwendung von Text eingesetzt werden kann.

So konnte gezeigt werden, dass der Zielkonflikt zwischen Sicherheit und Gebrauchstauglichkeit gelöst werden kann, wenngleich auch nur durch eine Erweiterung der Internetbrowser: Leider konnte mit dieser Studie die Sicherheit gegenüber Angriffen auf das HIP nicht belegt werden. Stattdessen wurde ein grundsätzliches Problem erkannt, welches seinen Ursprung in dem als Klartext vorliegenden Programmcode hat. Da dieses Problem generell für alle Webanwendungen besteht, stellt der gefundene Lösungsvorschlag eine weitreichende Verbesserung dar. Durch die hier vorgestellte Erweiterung der Internetbrowser könnte einerseits in vielen Bereichen auf HIPs verzichtet werden. Dazu gehören beispielsweise Kontaktformulare, die Erstellung von E-Mailkonten, Kommentaren oder Beiträgen in Foren, sowie der Schutz von Dateien, die bei Filehostern abgelegt wurden. Andererseits wäre dadurch ebenfalls eine Absicherung anderer Dienste gegen eine automatisierte Benutzung möglich. Leider konnte dieser Lösungsvorschlag nur konzeptuell getestet werden. Um darauf ein HIP-System aufzubauen ist in dieser Sache noch mehr Forschungsarbeit nötig. Auch in welcher Form dieses Verfahren auf andere Bereiche angewandt werden kann, muss im Einzelfall genau geprüft und gegebenenfalls erprobt werden.

Durch die Umfrage konnten Erkenntnisse zur Verbesserung des Prototyps gewonnen werden. Diese können in einem weiterführenden Prozess implementiert werden, sodass aus diesem Prototyp ein voll funktionsfähiges HIP-System entstehen könnte. Ein großes Hindernis auf diesem Weg ist derzeit noch die Schwachstelle des lesbaren Quellcodes, welche einen Angriff sehr einfach macht. Ob dieses Hindernis durch die hier vorgestellte Browsererweiterung beseitigt werden kann, ist fraglich. Denn diese Erweiterung müsste auf zahlreiche Browser angewendet werden um effektiv zu sein. Wer die Kosten für das benötigte Zertifikat tragen wird, ist derzeit ebenfalls ungeklärt.

Bei der Entwicklung eines HIPs sind sehr viele Aspekte zu berücksichtigen, die leider nicht alle mit dem erstellten Rahmenwerk abgedeckt werden können. Deshalb muss ein neues HIP immer einer genauen Prüfung hinsichtlich der Sicherheit und Gebrauchstauglichkeit unterzogen werden. Es gilt jedoch zu bedenken, dass es kein perfektes HIP geben kann, denn bereits die Existenz eines HIPs kann Benutzer stören. Deshalb lautet die Empfehlung für die Verwendung von HIPs, dass der Einsatz eines HIPs immer die letzte Option sein sollte. In einem solchen Fall sollte die Wahl der Technik auf ein spielbasiertes HIP fallen, da deren Akzeptanz bislang am größten ist. Zudem ist entscheidend, dass hiermit eine gute Gebrauchstauglichkeit und zugleich eine hohe Sicherheit erreicht werden kann. Der Einsatz von alternativen Eingabemöglichkeiten kann sehbehinderten Benutzern die Möglichkeit geben, diese HIPs ohne Hilfsmittel oder gar die Hilfe von Sehenden lösen zu können.

Ob in der Zukunft keine HIPs mehr verwendet werden, kann zu diesem Zeitpunkt nicht klar gesagt werden. Jedoch scheint sicher, dass immer mehr Sensoren für diesen Einsatz zur Verfügung stehen werden. Möglicherweise müssen HIPs dann nicht mehr aktiv vom Benutzer gelöst werden, sondern es entscheidet selbstständig durch die Auswertung von verschiedenen Sensoren ob der Kommunikationspartner ein Mensch oder eine Maschine ist.

Abbildungen

Tabellen

Literaturverzeichnis

[AFG10] Roland Austinat, Philipp Fechteler, and Hartmut Gieselmann. Über den Wolken - Wie Cloud Gaming den Spielemarkt revolutioniert. *c't*, (21):76f, 2010.

[Are14a] Are You A Human. Are You a Human - Fallstudie HiredMyWay. http://areyouahuman.com/case_studies/from-63-to-99-profile-completion/, Abgerufen am 15.01.2014, 2014.

[Are14b] Are You A Human. Are You a Human - Fallstudie VoteSane. http://areyouahuman.com/case_studies/spam-annihilated-with-playthru/, Abgerufen am 15.01.2014, 2014.

[Are14c] Are You A Human. Are You a Human - Homepage. http://areyouahuman.com/, Abgerufen am 15.01.2014, 2014.

[BAB+08] Jeffrey P Bigham, Maxwell B Aller, Jeremy T Brudvik, Jessica O Leung, Lindsay A Yazzolino, and Richard E Ladner. Inspiring blind high school students to pursue computer science with instant messaging chatbots. In *ACM SIGCSE Bulletin*, volume 40, pages 449–453. ACM, 2008.

[BBF+10] Elie Bursztein, Steven Bethard, Celine Fabry, John C. Mitchell, and Dan Jurafsky. How good are humans at solving captchas? a large scale evaluation. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, pages 399–413, Washington, DC, USA, 2010. IEEE Computer Society.

[BBP+11] E. Bursztein, R. Beauxis, H. Paskov, D. Perito, C. Fabry, and J. Mitchell. The failure of noise-based non-continuous audio captchas. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 19–31, 2011.

[BC09] Jeffrey P. Bigham and Anna C. Cavender. Evaluating existing audio captchas and an interface optimized for non-visual use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1829–1838, New York, NY, USA, 2009. ACM.

[Bri11] Harry Brignull. F**k captcha. http://www.90percentofeverything.com/2011/03/25/fk-captcha/, Abgerufen am 15.01.2014, März 2011.

[Chi13] Chip. Galaxy s4: Die wichtigsten sensoren des handys. http://www.chip.de/news/Galaxy-S4-Die-wichtigsten-Sensoren-des-Handys_61493678.html, Abgerufen am 11.02.2014, April 2013.

[CL96] Richard G Casey and Eric Lecolinet. A survey of methods and strategies in character segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(7):690–706, 1996.

[CLSC05a] Kumar Chellapilla, Kevin Larson, Patrice Simard, and Mary Czerwinski. Designing

human friendly human interaction proofs (hips). In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 711–720. ACM, 2005.

[CLSC05b] Kumar Chellapilla, Kevin Larson, Patrice Y Simard, and Mary Czerwinski. Computers beat humans at single character recognition in reading based human interaction proofs (hips). In *CEAS*, 2005.

[CS05] Kumar Chellapilla and Patrice Y. Simard. Using Machine Learning to Break Visual Human Interaction Proofs (HIPs). In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 265–272. MIT Press, Cambridge, MA, 2005.

[CT04] Monica Chew and JD Tygar. Image recognition captchas. *Computer Science*, 2004.

[D2113] Initiative D21. Mobile internetnutzung. http://www.initiatived21.de/wp-content/uploads/2013/02/studie_mobilesinternet_d21_huawei_2013.pdf, Abgerufen am 11.02.2014, Februar 2013.

[DDKN11] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From game design elements to gamefulness: Defining "gamification". In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, MindTrek '11, pages 9–15, New York, NY, USA, 2011. ACM.

[DPY12] D. D'Souza, P.C. Polina, and R.V. Yampolskiy. Avatar captcha: Telling computers and humans apart via face classification. In *Electro/Information Technology (EIT), 2012 IEEE International Conference on*, pages 1–6, 2012.

[EDHS07] Jeremy Elson, John R Douceur, Jon Howell, and Jared Saul. Asirra: a captcha that exploits interest-aligned manual image categorization. In *ACM Conference on Computer and Communications Security*, pages 366–374, 2007.

[Fun] FunCAPTCHA. Stop spam with a free, fun, fast captcha game. http://www.funcaptcha.co/, Abgerufen am 17.02.2014.

[GG06] Alan Grosskurth and Michael W Godfrey. Architecture and evolution of the modern web browser. *Preprint submitted to Elsevier Science*, 2006.

[Gol08] Philippe Golle. Machine learning attacks against the asirra captcha. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, CCS '08, pages 535–542, New York, NY, USA, 2008. ACM.

[GWF12] Haichang Gao, Wei Wang, and Ye Fan. Divide and conquer: An efficient attack on yahoo! captcha. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, pages 9–16. IEEE, 2012.

[GWLY11] Feng Guo, Miao Wang, Yanwei Li, and Hongyan Yan. The vulnerabilities and status of captchas. In *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, pages 448–452, 2011.

[Ham13] Juho Hamari. Transforming homo economicus into homo ludens: A field experiment on gamification in a utilitarian peer-to-peer trading service. *Electronic Commerce Research*

and Applications, 12(4):236–245, 2013.

[HCR09] Carlos Javier Hernandez-Castro and Arturo Ribagorda. Remotely telling humans and computers apart: An unsolved problem. In *iNetSec 2009–Open Research Problems in Network Security*, pages 9–26. Springer, 2009.

[HCR10] Carlos Javier Hernandez-Castro and Arturo Ribagorda. Pitfalls in captcha design and implementation: The math captcha, a case study. *computers & security*, 29(1):141–157, 2010.

[HCRS09] Carlos Javier Hernandez-Castro, Arturo Ribagorda, and Yago Saez. Side-channel attack on labeling captchas. *arXiv preprint arXiv:0908.1185*, 2009.

[HGH08] Abram Hindle, Michael W Godfrey, and Richard C Holt. Reverse engineering captchas. In *Reverse Engineering, 2008. WCRE'08. 15th Working Conference on*, pages 59–68. IEEE, 2008.

[HHY12] Kuo-Feng Hwang, Cian-Cih Huang, and Geeng-Neng You. A spelling based captcha system by using click. In *Biometrics and Security Technologies (ISBAST), 2012 International Symposium on*, pages 1–8, 2012.

[HKS14] J. Hamari, J. Koivisto, and H. Sarsa. Does gamification work? – a literature review of empirical studies on gamification. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on*, pages 3025–3034, Jan 2014.

[HLBO10] Shih-Yu Huang, Yeuan-Kuen Lee, Graeme Bell, and Zhan-he Ou. An efficient segmentation algorithm for captchas with line cluttering and character warping. *Multimedia Tools and Applications*, 48(2):267–289, 2010.

[JS12] Sowmya Jain and B. Sivaselvan. Usability aspects of hci in the design of captchas. In *Computational Intelligence Computing Research (ICCIC), 2012 IEEE International Conference on*, pages 1–4, 2012.

[Kal12] Gursev Kalra. Captcha re-riding attack. http://blog.opensecurityresearch.com/2012/02/captcha-re-riding-attack.html, Abgerufen am 23.01.2014, Februar 2012.

[KLS02] Greg Kochanski, Daniel Lopresti, and Chilin Shih. A reverse turing test using speech. In *Seventh International Conference on Spoken Language Processing*, 2002.

[Klu08a] Kurt Alfred Kluever. Breaking the paypal hip: A comparison of classifiers. Technical report, Department of Computer Science Rochester Institute of Technology, 2008.

[Klu08b] Kurt Alfred Kluever. Character segmentation and classification. Technical report, Department of Computer Science Golisano College of Computing and Information Sciences Rochester Institute of Technology February 28, 2008, 2008.

[Klu08c] Kurt Alfred Kluever. Evaluating the usability and security of a video captcha. Master's thesis, Rochester Institute of Technology, Rochester, NY, USA, August 2008.

[KMCY12] M. Korayem, A.A. Mohamed, D. Crandall, and R.V. Yampolskiy. Learning visual features for the avatar captcha recognition challenge. In *Machine Learning and Applications*

(ICMLA), 2012 11th International Conference on, volume 2, pages 584–587, 2012.

[KMD13] Immanuel Koenig, Abdul Qudoos Memon, and Klaus David. Energy consumption of the sensors of smartphones. In *Wireless Communication Systems (ISWCS 2013), Proceedings of the Tenth International Symposium on*, pages 1–5, 2013.

[Kot04] Munir Kotadia. Porn gets spammers past hotmail, yahoo barriers. http://news.cnet.com/2100-1023-5207290.html, Abgerufen im Januar 2014, 2004.

[LABB98] M.D. Lillibridge, M. Abadi, K. Bharat, and A.Z. Broder. Method for selectively restricting access to computer systems, 1998. US Patent 6,195,698.

[LHBL11] Rosa Lin, Shih-Yu Huang, Graeme B Bell, and Yeuan-Kuen Lee. A new captcha interface design for mobile devices. In *Proceedings of the Twelfth Australasian User Interface Conference-Volume 117*, pages 3–8. Australian Computer Society, Inc., 2011.

[Lin] Linkcrypt. Willkommen auf linkcrypt.ws. http://linkcrypt.ws/, Abgerufen am 03.02.2014.

[LSK+10] Shujun Li, S Shah, M Khan, Syed Ali Khayam, Ahmad-Reza Sadeghi, and Roland Schmitz. Breaking e-banking captchas. In *Proceedings of the 26th Annual Computer Security Applications Conference*, pages 171–180. ACM, 2010.

[Mar11] Eldar Marcussen. Captcha reload and other attacks. http://www.justanotherhacker.com/2011/12/captcha-reload-attack.html, Abgerufen am 23.01.2014, Dezember 2011.

[MJHP04] Gabriel Moy, Nathan Jones, Curt Harkless, and Randall Potter. Distortion estimation techniques in solving visual captchas. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–23. IEEE, 2004.

[MM03] Greg Mori and Jitendra Malik. Recognizing objects in adversarial clutter: Breaking a visual captcha. In *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR'03, pages 134–141, Washington, DC, USA, 2003. IEEE Computer Society.

[MMH09] J.P. McIntire, L.K. McIntire, and P.R. Havig. A variety of automated turing tests for network security: Using ai-hard problems in perception and cognition to ensure secure collaborations. In *Collaborative Technologies and Systems, 2009. CTS '09. International Symposium on*, pages 155–162, 2009.

[Mot13] David Moth. Six alternatives to using the dreaded captcha images. http://econsultancy.com/blog/63144-six-alternatives-to-using-the-dreaded-captcha-images, Abgerufen am 23.02.2014, July 2013.

[Moz] Mozilla Developer Network. Detecting device orientation. https://developer.mozilla.org/de/docs/WebAPI/Detecting_device_orientation, Abgerufen am 19.03.1985.

[MSG+13] Manar Mohamed, Niharika Sachdeva, Michael Georgescu, Song Gao, Nitesh Saxena, Chengcui Zhang, Ponnurangam Kumaraguru, Paul C van Oorschot, and Wei-Bang Chen. Three-way dissection of a game-captcha: Automated attacks, relay attacks, and usability. *arXiv preprint arXiv:1310.1540*, 2013.

[NB12] N. Nanglae and P. Bhattarakosol. A study of human bio-detection function under text-based captcha system. In *Computer and Information Science (ICIS), 2012 IEEE/ACIS 11th International Conference on*, pages 139–144, 2012.

[OM12] M. Okada and S. Matsuyama. New captcha for smartphones and tablet pc. In *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*, pages 34–35, 2012.

[Pho] PhoneGap. Homepage. http://phonegap.com/about/feature/, Abgerufen am 18.03.2014.

[PK05] C. Pope and Khushpreet Kaur. Is it human or computer? defending e-commerce with captchas. *IT Professional*, 7(2):43–49, 2005.

[Pon06] M. Ponec. Visual reverse turing tests: A false sense of security. In *Information Assurance Workshop, 2006 IEEE*, pages 305–311, 2006.

[Pre] Premohit. Kugellabyrinth. http://www.premohit.ch/werbeartikel/spielzeug-spielwaren.html, Abgerufen am 14.02.2014.

[Pur05] PureMango. Breaking captchas without using ocr. http://www.puremango.co.uk/2005/11/breaking_captcha_115/, Abgerufen im Januar 2014, 2005.

[RC13] Gerardo Reynaga and Sonia Chiasson. The usability of captchas on smartphones. In Pierangela Samarati, editor, *SECRYPT*, pages 427–434. SciTePress, 2013.

[reC04] reCAPTCHA. What is recaptcha? http://www.google.com/recaptcha/learnmore Abgerufen am 11.01.2014, 2004.

[RTD12] Rahman, Tomar, and Das. Dynamic image based captcha. In *Communication Systems and Network Technologies (CSNT), 2012 International Conference on*, pages 90–94, 2012.

[SEAYM10] Ahmad Salah El Ahmad, Jeff Yan, and Lindsay Marshall. The robustness of a new captcha. In *Proceedings of the Third European Workshop on System Security*, pages 36–41. ACM, 2010.

[SHFL08] Graig Sauer, Harry Hochheiser, Jinjuan Feng, and Jonathan Lazar. Towards a universally usable captcha. In *Proc. of the 4th Symp. On Usable Privacy and Security (SOUPS'08), Pittsburgh, PA, USA*, 2008.

[SPA12] SPAMtech. Cracking the areyouahuman captcha. http://spamtech.co.uk/software/bots/cracking-the-areyouhuman-captcha/, Abgerufen am 16.01.2014, 2012.

[SSB+03] Patrice Y Simard, Richard Szeliski, Josh Benaloh, Julien Couvreur, and Iulian Calinov. Using character recognition and segmentation to tell computer from humans. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pages 418–423. IEEE, 2003.

[SSSS06] M. Shirali-Shahreza and S. Shirali-Shahreza. Drawing captcha. In *Information Technology Interfaces, 2006. 28th International Conference on*, pages 475–480, 2006.

[Sur] W3Techs World Wide Web Technology Surveys. Server-side programming languages.

http://w3techs.com/, Abgerufen am 13.01.2014.

[Sut12] Cody Sutherland. Usability and security of text-based captchas. In *UMM CSci Senior Seminar Conference, Morris, MN*, 2012.

[Tha13] E. Thammasat. The statistical recognition of walking, jogging, and running using smartphone accelerometers. In *Biomedical Engineering International Conference (BMEiCON), 2013 6th*, pages 1–4, Oct 2013.

[TSHA08] Jennifer Tam, Jiri Simsa, Sean Hyde, and Luis V Ahn. Breaking audio captchas. In *Advances in Neural Information Processing Systems*, pages 1625–1632, 2008.

[Tul] Rob Tuley. Do you really need a captcha? http://textcaptcha.com/really, Abgerufen am 23.02.2014.

[Tur50] Alan M Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[Vrb13] *The evolution of web browser architecture*, volume 1. EDIS - Publishing Institution of the University of Zilina, 2013.

[W3C] W3C. Deviceorientation event specification. http://w3c.github.io/deviceorientation/spec-source-orientation.html, Abgerufen am 13.01.2014.

[Wag] Wagenschenke. Homerun. http://www.wagenschenke.ch/HomeRun.swf, Abgerufen am 14.02.2014.

[Wik] Wikipedia. 15-puzzle. http://de.wikipedia.org/wiki/15-Puzzle, Abgerufen am 10.02.2014.

[WJL12] Te-En Wei, A.B. Jeng, and Hahn-Ming Lee. Geocaptcha - a novel personalized captcha using geographic concept to defend against 3rd party human attack. In *Performance Computing and Communications Conference (IPCCC), 2012 IEEE 31st International*, pages 392–399, 2012.

[Yan06] Jeff Yan. Security protocols. In Bruce Christianson, Bruno Crispo, James A. Malcolm, and Michael Roe, editors, *Security Protocols*, chapter Bot, Cyborg and Automated Turing Test, pages 190–197. Springer-Verlag, Berlin, Heidelberg, 2006.

[YC12a] Toshihiko Yamasaki and Tsuhan Chen. Face recognition challenge: Object recognition approaches for human/avatar classification. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, pages 574–579, 2012.

[YC12b] Toshihiko Yamasaki and Tsuhan Chen. Spatial statistics for spatial pyramid matching based image recognition. In *Signal Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*, pages 1–10, 2012.

[YSEA07] Jeff Yan and Ahmad Salah El Ahmad. Breaking visual captchas with naive pattern recognition algorithms. In *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, pages 279–291. IEEE, 2007.

[YSEA08a] Jeff Yan and Ahmad Salah El Ahmad. A low-cost attack on a microsoft captcha. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*,

CCS '08, pages 543–554, New York, NY, USA, 2008. ACM.

[YSEA08b] Jeff Yan and Ahmad Salah El Ahmad. Usability of captchas or usability issues in captcha design. In *Proceedings of the 4th Symposium on Usable Privacy and Security*, SOUPS '08, pages 44–52, New York, NY, USA, 2008. ACM.

[YSEA09] Jeff Yan and Ahmad Salah El Ahmad. Captcha security: a case study. *Security & Privacy, IEEE*, 7(4):22–28, 2009.

7 Anhang

Quellcode 7.1: index.php

```
<!DOCTYPE html>
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<title>Human Interaction Proof</title>
<meta name="viewport" content="width=210, initial-scale=1">
<script src="SecureHttpRequest.js"></script>
<script src="Attacks.js"></script>
<?php
session_start();
//define canvas and image size
$size = 200;
$marble = array ();
$hole = array ();
$line = array ();
$marble["r"] = 5;
$hole["r"] = 7;
//get random positions of the hole and marble
$hole["x"] = mt_rand(0 + 2*$hole["r"], $size - 2*$hole["r"]);
$hole["y"] = mt_rand(0 + 2*$hole["r"], $size - 2*$hole["r"]);
$marble["x"] = mt_rand(0 + $marble["r"], $size - $marble["r"]);
$marble["y"] = mt_rand(0 + $marble["r"], $size - $marble["r"]);
while (abs($marble["x"] - $hole["x"]) < $size/6)
{
    $marble["x"] = mt_rand(0 + $marble["r"], $size - $marble["r"]);
}
$line["x"] = ($hole["x"] < $marble["x"] ?
              mt_rand($hole["x"] + 2*$hole["r"], $marble["x"] - $marble["r"]) :
              mt_rand($marble["x"] + $marble["r"], $hole["x"] - 2*$hole["r"]));
if ( mt_rand ( -2, 2 ) > 0 )
{
    $line["y_min"] = 0;
    $line["y_max"] = $size - $marble["r"] * 2.1;
}
else
{
    $line["y_min"] = $marble["r"] * 2.1;
    $line["y_max"] = $size;
}

$_SESSION["time"] = time();
$_SESSION["line"] = array ($line["x"], $line["y_min"], $line["y_max"]);
$_SESSION["marble"] = array ($marble["x"], $marble["y"]);
$_SESSION["hole"] = array ($hole["x"], $hole["y"]);
$_SESSION["result"] = null;
//Write the Games JavaScript
$js = "<script>
      var size = ".$size.";
      var marble_r = ".$marble["r"].";
      var marble_x = m_x = ".$marble["x"].";
      var marble_y = m_y = ".$marble["y"].";
      var hole_r = ".$hole["r"].";
      var hole_x = ".$hole["x"].";
      var hole_y = ".$hole["y"].";
      var line_x = ".$line["x"].";
      var line_y_min = ".$line["y_min"].";
      var line_y_max = ".$line["y_max"].";".
      file_get_contents ("HIP.js").
      "</script>";
$_SESSION["hash"] = hash ("sha512", $js);
```

```
60 echo $js;
61 ?>
62
63 <style>
64 body {
65     font-family: Arial, Helvetica, sans-serif;
66 }
67 a {
68     color: black;
69     text-decoration: none;
70 }
71 </style>
72 </head>
73 <body>
74     <div id="help" style="display: none; position: absolute; height: 100%; background-color: white;" >
75        <a href="index.php" ><img src="Bilder/Anleitung.png" /></a></div>
76     <div style="background: green; width: 200px; margin-bottom: 3px;" id="time"> </div>
77     <canvas style="border: 1px solid #000;" id="HIP" width="200" height="200"></canvas>
78     <div style="width:200px; text-align: right;"><a href="#" onclick="showHelp();" >?</a></div>
79     <a href="#" onclick="attack1()"><p>Angriff 1 testen</p></a>
80     <a href="#" onclick="attack2()"><p>Angriff 2 testen</p></a>
81 </body>
82 </html>
```

Quellcode 7.2: HIP.js

```
1  var canvas = context = drawInterval = null;
2  var data = new Array ();
3  var frame = acc_x = acc_y = tmp_x = tmp_y = v_x = v_y = 0;
4  var time = 10.0;
5  var time_div;
6
7  window.onload = function ()
8  {
9      canvas = document.getElementById ("HIP");
10     time_div = document.getElementById ("time");
11     if (window.DeviceOrientationEvent)
12     {
13         context = canvas.getContext("2d");
14         data.push ( [marble_x, marble_y] );
15         data.push ( [hole_x, hole_y] );
16         drawInterval = setInterval(function(){draw();}, 40);
17         window.addEventListener("compassneedscalibration", function(event)
18         {event.preventDefault();}, true);
19         window.addEventListener("deviceorientation", function(event)
20         {
21             if ( window.innerHeight > window.innerWidth )
22             {
23                    acc_x = Math.round (event.gamma);
24                    acc_y = Math.round (event.beta);
25             }
26             else
27             {
28                    acc_x = Math.round (event.beta);
29                    acc_y = Math.round (event.gamma) * -1;
30             }
31         }, true);
32     }
33     else
34     {
35         canvas.style.display = "none";
36         time_div.style.background = "white";
37         time_div.style.position = "absolute";
38         time_div.style.height = "100%";
39         time_div.innerText = "Kein Sensor vorhanden!";
40     }
41 };
42 function draw ( )
43 {
44     settime ();
```

```
45    if (checkCollision ())
46    {
47        canvas.style.background = "green";
48        data.push ( frame, time );
49        clearInterval ( drawInterval );
50        endGame();
51    }
52    context.clearRect ( 0, 0, size, size );
53    context.lineWidth = 1;
54    //draw barrier
55    context.beginPath();
56    context.moveTo(line_x,line_y_min);
57    context.lineTo(line_x,line_y_max);
58    context.stroke();
59    //draw hole
60    context.beginPath();
61    context.arc(hole_x, hole_y, hole_r, 0, 2 * Math.PI);
62    context.fillStyle = 'black';
63    context.fill();
64    //draw marble
65    context.beginPath();
66    context.arc(marble_x, marble_y, marble_r, 0, 2 * Math.PI);
67    context.fillStyle = 'red';
68    context.fill();
69    frame += 1;
70}
71function settime ()
72{
73    time = time - 0.04;
74    time_div.style.width = Math.max(Math.floor (20 * time),0) + "px";
75    if ( time <= 0)
76    {
77        canvas.style.background = "red";
78        clearInterval ( drawInterval );
79        window.location.replace(window.location.pathname);
80    }
81    else if ( time < 5 && time > 2 ) { time_div.style.background = "orange"; }
82    else if ( time < 2 ) { time_div.style.background = "red"; }
83}
84function checkCollision ()
85{
86    if ( tmp_x != acc_x || tmp_y != acc_y ) { data.push ( [frame, acc_x, acc_y] ); }
87    tmp_x = acc_x;
88    tmp_y = acc_y;
89    v_x = ( marble_x - marble_r <= 0 || marble_x + marble_r >= size ?
90              v_x * -0.5 + tmp_x * 0.1:
91              v_x * 0.9 + tmp_x * 0.1);
92    v_y = ( marble_y - marble_r <= 0 || marble_y + marble_r >= size ?
93              v_y * -0.5 + tmp_y * 0.1:
94              v_y * 0.9 + tmp_y * 0.1);
95    if ( marble_y > line_y_min && marble_y < line_y_max &&
96         ( ( v_x > 0 && marble_x + marble_r <= line_x && marble_x + marble_r + v_x > line_x ) ||
97         ( v_x < 0 && marble_x - marble_r >= line_x && marble_x - marble_r + v_x < line_x ) ) )
98    {
99        marble_x = ( v_x > 0 ? line_x - marble_r : line_x + marble_r );
100       v_x = v_x * -0.5;
101   }
102   else
103   {
104       marble_x = Math.min ( Math.max ( marble_x + v_x, 0 + marble_r ), size - marble_r );
105   }
106   if ( marble_x - marble_r < line_x && marble_x + marble_r > line_x )
107   {
108       v_y = 0;
109   }
110   marble_y = Math.min ( Math.max ( marble_y + v_y, 0 + marble_r ), size - marble_r );
111   var dist = Math.round(Math.sqrt(Math.pow(marble_x - hole_x, 2) + Math.pow(marble_y - hole_y, 2)));
112   return dist < marble_r;
113}
114function endGame ()
```

```
{
    var request = new SecureHttpRequest();
    request.addEventListener("readystatechange", function ()
        {
            if( this.readyState === 4 ) { window.location = "result.php";}
        });
    request.send("POST", "check.php", true, JSON.stringify(data));
}
function showHelp ()
{
    clearInterval ( drawInterval );
    document.getElementById ( "help" ).style.display = "block";
}
```

Quellcode 7.3: SecureHttpRequest.js

```
function SecureHttpRequest() {

    this.request = null;
    // Mozilla
    if (window.XMLHttpRequest) {
        this.request = new XMLHttpRequest();
    }
    // IE
    else if (window.ActiveXObject) {
        this.request = new ActiveXObject("Microsoft.XMLHTTP");
    }
    this.addEventListener = function (event, callback)
        {
            this.request.addEventListener ( event, callback);
        };
    this.send = function (method, action, async, data)
        {
            //This should be done by the Browser in the future
            //********************
            //data = AES.encrypt(SHA512.encrypt(javaScriptCode) + ";" + data, BrowsersPrivateKey);
            //********************
            this.request.open (method, action, async);
            this.request.setRequestHeader("Content-type","application/x-www-form-urlencoded");
            this.request.send("data="+data);
        };
}
```

Quellcode 7.4: check.php

```
<!DOCTYPE html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=yes" />
<style>
    body {
        font-family: Arial, Helvetica, sans-serif;
    }
    a {
        color: black;
        text-decoration: none;
    }
</style>
</head>
<body>
<?php
if ( !isset ($_POST["data"])) die;
session_start();

$time = time();

//Can only be done in the future (see SecureHttpRequest.js)
/*
 * $data = AES.decrypt($_POST["data"], BrowsersPublicKey);
 * if ( strpos($data, $_SESSION["hash"]) !== 0 )
```

```
25 * {
26 * $_SESSION["result"] = false;
27 * die;
28 * }
29 * $data = json_decode (substr ($data, strlen ($_SESSION["hash"])));
30 */
31//Instead do only this
32$data = json_decode($_POST["data"]);
33
34$timejs = array_pop ( $data );
35$frames = array_pop ( $data );
36$marble = array_shift ( $data );
37$hole = array_shift ( $data );
38$variance = array_unique(array_merge(array_map(function ($item){ return $item[1]; }, $data),
39                            array_map(function ($item){ return $item[2]; }, $data)));
40$dist = 9999;
41
42if ( $_SESSION["marble"] == $marble && //Stimmen die Startkoordinaten überein?
43     $_SESSION["hole"] == $hole && //Stimmen die Startkoordinaten überein?
44     $time - $_SESSION["time"] < 15 && //Sind nicht mehr als 15 Sekunden vergangen?
45     $timejs > 0 && //Wurde das Spiel innerhalb von 10 Sekunden gelöst?
46     count ($data) > 7 && //Wurden mindestens 7 Messwerte geliefert?
47     $frames > 50 && //Wurde das Spiel mindestens 2 Sekunden gespielt?
48     $time - $_SESSION["time"] > 2 && //Wurde das Spiel mindestens 2 Sekunden gespielt?
49     count ($variance) > 5 ) //Weisen die Daten eine Varianz größer 5 auf?
50{
51    //Gesammelte Daten prüfen
52    $size = 200;
53    $marble_r = 5;
54    $marble_x = $_SESSION["marble"][0];
55    $marble_y = $_SESSION["marble"][1];
56    $hole_x = $_SESSION["hole"][0];
57    $hole_y = $_SESSION["hole"][1];
58    $line_x = $_SESSION["line"][0];
59    $line_y_min = $_SESSION["line"][1];
60    $line_y_max = $_SESSION["line"][2];
61    $acc_x = $acc_y = $v_x = $v_y = $j = 0;
62    for ( $i = 0; $i <= $frames; $i++ )
63    {
64        if ( $data[$j][0] == $i )
65        {
66            $acc_x = $data[$j][1];
67            $acc_y = $data[$j][2];
68            $j++;
69        }
70        $v_x = ( $marble_x - $marble_r <= 0 || $marble_x + $marble_r >= $size ?
71                  $v_x * -0.5 + $acc_x * 0.1:
72                  $v_x * 0.9 + $acc_x * 0.1);
73        $v_y = ( $marble_y - $marble_r <= 0 || $marble_y + $marble_r >= $size ?
74                  $v_y * -0.5 + $acc_y * 0.1:
75                  $v_y * 0.9 + $acc_y * 0.1);
76        if ( $marble_y > $line_y_min && $marble_y < $line_y_max &&
77             (($v_x > 0 && $marble_x + $marble_r <= $line_x && $marble_x + $marble_r + $v_x > $line_x)||
78             ($v_x < 0 && $marble_x - $marble_r >= $line_x && $marble_x - $marble_r + $v_x < $line_x)))
79        {
80            $marble_x = ( $v_x > 0 ? $line_x - $marble_r : $line_x + $marble_r );
81            $v_x = $v_x * -0.5;
82        }
83        else
84        {
85            $marble_x = min ( max ( $marble_x + $v_x, 0 + $marble_r ), $size - $marble_r );
86        }
87        if ( $marble_x - $marble_r < $line_x && $marble_x + $marble_r > $line_x )
88        {
89            $v_y = 0;
90        }
91        $marble_y = min ( max ( $marble_y + $v_y, 0 + $marble_r ), $size - $marble_r );
92    }
93    $dist = round ( sqrt ( pow( $marble_x - $hole_x, 2 ) + pow ( $marble_y - $hole_y, 2 ) ), 0 );
94    $_SESSION["result"] = $dist < $marble_r;
```

```
95 }
96 else
97 {
98     $_SESSION["result"] = false;
99 }
100 ?>
101 </body>
102 </html>
```

Quellcode 7.5: result.php

```
1 <!DOCTYPE html>
2 <head>
3 <meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=yes" />
4 <style>
5     body {
6         font-family: Arial, Helvetica, sans-serif;
7     }
8     a {
9         color: black;
10        text-decoration: none;
11    }
12 </style>
13 </head>
14 <body>
15 <?php
16 session_start();
17
18 while ( $_SESSION["result"] === null ) {}
19 if ( $_SESSION["result"] === true )
20 {
21     echo "<p>Gut gemacht!</p>";
22     //Destroy session
23     $_SESSION = array();
24     if (ini_get("session.use_cookies"))
25     {
26         $params = session_get_cookie_params();
27         setcookie(session_name(), '', time() - 42000,
28                 $params["path"], $params["domain"],
29                 $params["secure"], $params["httponly"]);
30     }
31     session_destroy();
32 }
33 else
34 {
35     echo "<p>Bitte versuchen Sie es nocheinmal.</p>";
36 }
37 ?>
38 <a href="index.php"><p><= Nochmal</p></a>
39 </body>
40 </html>
```

Quellcode 7.6: Attacks.js

```
1 function attack1 ()
2 {
3     clearInterval ( drawInterval );
4     marble_x = m_x;
5     marble_y = m_y;
6     data = new Array ();
7     data.push ( [marble_x, marble_y] );
8     data.push ( [hole_x, hole_y] );
9     frame = 1;
10    time = 10;
11    drawInterval = setInterval(function(){attack1move();}, 40);
12 }
13 function attack1move ()
14 {
15    if ( time > 1 )
```

```
    {
        acc_x = rand()-5;
        acc_y = rand()-5;
        draw();
    }
    else
    {
        clearInterval ( drawInterval );
        data.push ( frame, time );
        endGame();
    }
}
function attack2 ()
{
    clearInterval ( drawInterval );
    marble_x = m_x;
    marble_y = m_y;
    data = new Array ();
    data.push ( [marble_x, marble_y] );
    data.push ( [hole_x, hole_y] );
    frame = 1;
    time = 10;
    drawInterval = setInterval(function(){attack2move1();}, 40);
}
function attack2move1 ()
{
    //roll marble up or down and sideways
    if ( ( marble_y > marble_r &&
          marble_y < size - marble_r ) ||
          Math.abs (marble_x - hole_x) > marble_r )
    {
        acc_x = ( marble_x > hole_x ? -rand() : rand());
        acc_y = ( line_y_min == 0 ? rand() : -rand());
        draw ();
    }
    else
    {
        clearInterval ( drawInterval );
        drawInterval = setInterval(function(){attack2move2();}, 40);
    }
}
function attack2move2 ()
{
    //roll the marble in the hole
    acc_x = ( marble_x > hole_x ? -rand() : rand());
    acc_y = ( marble_y > hole_y ? -rand() : rand());
    draw();
}
function rand ( )
{
    return Math.floor(Math.random()*10);
}
```